ChipScope ILA Software and Cores User Manual

(ChipScope Software v4.2i)

UG005 / PN 0401884 (v4.2) March 22, 2002

XILINX®

The Xilinx logo shown above is a registered trademark of Xilinx, Inc.

The shadow X shown above is a trademark of Xilinx, Inc.

The following table shows the revision history for this document.

	Version	Revision
03/06/00	1.0	Initial Xilinx release.
06/30/00	1.1	Deleted ILA Core 0.5 and Earlier section; Added Parallel Cable III references
12/15/00	2.0	Removed Tutorial (old Chapter 4); Added Using the ChipScope Analyzer (new Chapter 4); Defined ChipScope Tools and its components
08/10/01	3.0	Imported manual into FrameMaker v6.0. Added Signal Name Importing section; added Software Requirements section; added Project Level Parameters section; added Refresing the Netlist section; replaced Installing ChipScope Tools section with ChipScope Software Installation section; replaced Host System Requirements with two new sections: 1) Requirements for Host Systems Running Windows and 2)Requirements for Host Systems Running Solaris 5.6; updated System Requirements section; updated Software Installation section; updated screen shots.
08/27/01	3.1	Added software version to title page; miscellaneous non-technical text edits.
10/19/01	4.0	Changed Alliance Series 3.1i to Xilinx ISE 4.1i. Changed Xilinx Foundation Series 3.1i to Xilinx ISE Foundation 4.1i. In Chapters 2 and 3, converted Data Depth section text to tables. Added Fig 4-11 and Fig 4-12. Replaced Fig 4-13. Miscellaneous edits for clarification.
03/22/02	4.2	Updated text, graphics, and screen shots from 4.1i to 4.2i compatibility.

Contents

Chapter 3: Using the ChipScope Core Inserter

Chapter 4: Using the ChipScope Analyzer

Chapter 1

Introduction

ChipScope ILA Tools Overview

As the density of FPGA devices increases, so does the impracticality of attaching test equipment probes to these devices under test. The ChipScope™ ILA tools integrate key logic analyzer hardware components with the target design inside the Virtex™ device. The ChipScope ILA tools communicate with these components and provide the designer with a complete logic analyzer, without the need for cumbersome probes or expensive test equipment.

ChipScope ILA Tools Description

Table 1-1: **ChipScope ILA Tools Description**

Tool	Description
ChipScope Core Generator	Provides netlists and instantiation templates for the Integrated CONtroller (ICON) core and the Integrated Logic Analyzer (ILA) core.
ChipScope Core Inserter	Automatically inserts the ICON core and the ILA core into the user's synthesized design.
ChipScope Analyzer	Allows setup and trace display for the ILA core. The ILA core provides the trigger and trace capture capability. The ICON core communicates to the dedicated Boundary Scan pins.

The ChipScope Analyzer supports the Xilinx MultiLINX™, Parallel Cable III and Parallel Cable IV download cables for communication between the PC and FPGA(s). The MultiLINX cable supports both USB (Windows 98 and Windows 2000) and RS-232 serial communication from the PC (see Figure 1-1, page 2). The Parallel Cable III and Parallel Cable IV cables support only parallel port communication from the PC to the Boundary Scan chain.

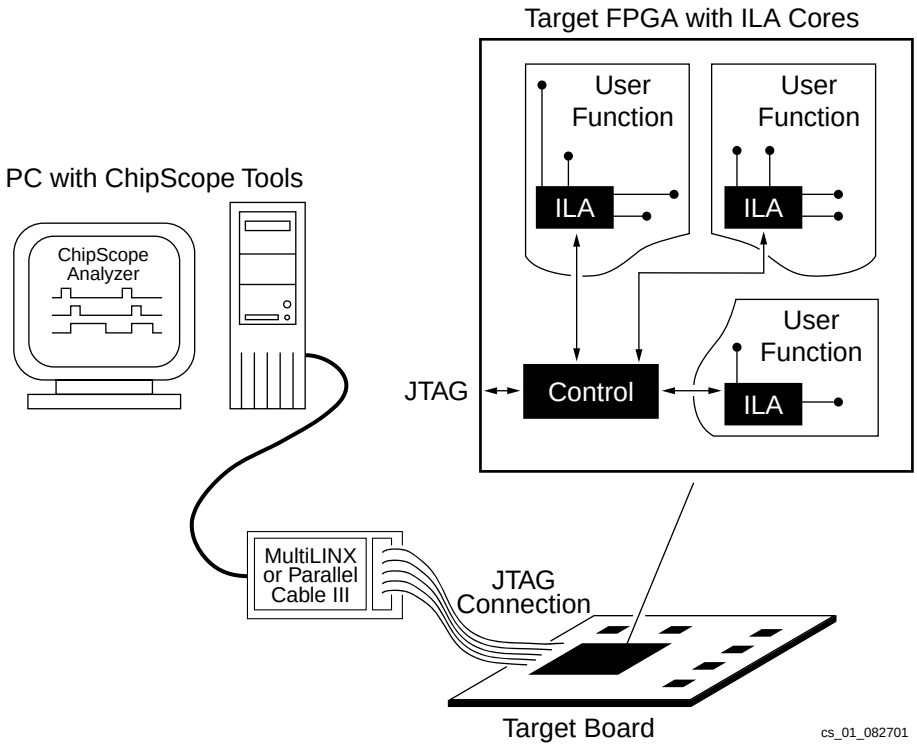

Figure 1-1: **ChipScope Block Diagram**

Users can place the ILA and ICON cores into their design in one of two ways:

- By generating the cores with the ChipScope Core Generator and instantiating them into the source HDL code
- By inserting the cores into the post-synthesis EDIF netlist using the ChipScope Core Inserter

The design is then placed and routed using the Xilinx ISE Alliance 4.2i (or later) or Xilinx ISE Foundation 4.2i (or later) implementation tools. Next, the user downloads the bitstream and analyzes the design with the ChipScope Analyzer software.

The ChipScope Analyzer contains many features that Xilinx FPGA designers need for thoroughly verifying their logic (Table 1-2). User-selectable data channels range from 1 to 256, and the number of sample sizes ranges from 256 to 16384, effectively doubling any FPGA logic analysis capability on the market today. Users can change the triggers in real time without affecting their logic. The easy-to-use ChipScope Analyzer leads designers through the process of modifying triggers and analyzing the data.

Table 1-2: **ChipScope Features and Benefits**

Feature	Benefit
1 to 256 user-selectable data channels	Accurately captures wide data bus functionality
User-selectable sample buffers ranging in size from 256 to 16384 samples	Large sample size increases accuracy and probability of capturing infrequent events
Separate bus trigger with user-selectable width of 1-64 bits	Separate trigger bus reduces need for sample storage
All data and trigger operations are synchronous to the user clock up to 155 MHz	Capable of high-speed data capture
Trigger conditions are in-system changeable without affecting the user logic	No need to single step or stop a design for logic analysis
Can write waveforms to VCD, FBDF, and ASCII formats	Compatible with Agilent Technologies and other waveform viewers
Easy-to-use graphical interface	Guides users through selecting the correct options
Up to 15 independent ILA capture cores per device	Can segment logic and test smaller sections of a large design for greater accuracy
Multiple trigger settings	Records duration and number of events along with matches and ranges for greater accuracy and flexibility
Downloadable from the Xilinx Web site	Tools are easily accessible from the ChipScope Suite

Design Flow

The ChipScope Tools design flow (Figure 1-2) merges easily with any standard FPGA design flow that uses a standard HDL synthesis tool and the Xilinx ISE Alliance 4.2i (or later) or Xilinx ISE Foundation 4.2i (or later) implementation tools.

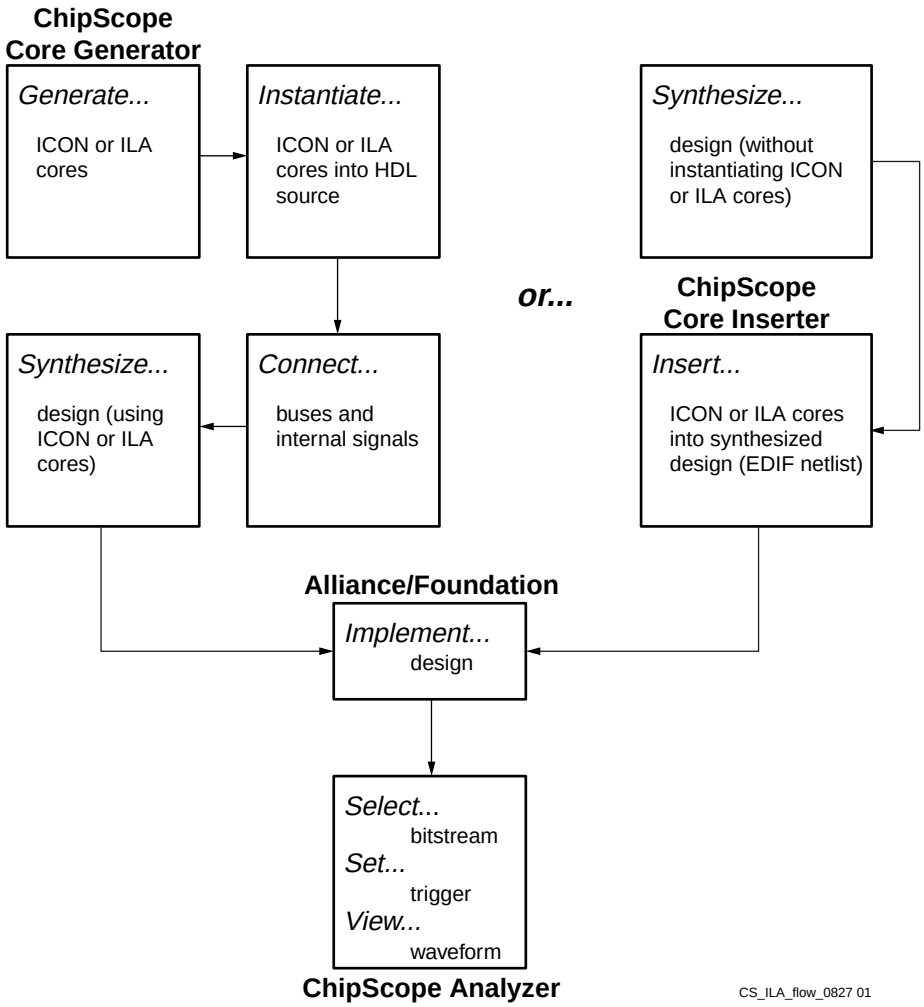

Figure 1-2: **ChipScope Tools Design Flow**

Trigger Settings

The ILA core has two trigger modes: *basic* and *extended*. The basic trigger mode provides up to two trigger match units capable of detecting a single exact match for every trigger condition. The extended trigger mode adds the ability to do range matching, trigger pulse width duration measurement, trigger event counting, and *if-then* trigger macros. Table 1-3 compares the basic and extended trigger modes.

Table 1-3: **Basic and Extended ILA Trigger Modes**

Trigger Mode Feature	Basic Mode	Extended Mode
Up to two match units	Yes	Yes
Trigger function combining all match units	Yes	Yes
Match value and edge comparison	Yes	Yes
Match range comparison (such as >, ≥ <,≤)	No	Yes
Trigger pulse duration measurement	No	Yes
Trigger event count measurement	No	Yes
If-then trigger macros	No	Yes

External Trigger Description

The ChipScope Analyzer software can accept a trigger input signal and generate a trigger output signal for use with external test equipment.

An external trigger input signal must enter the device on a normal input pin connected to the ICON core unit where it is distributed to each of 15 possible ILA components in the design.

Similarly, each ILA core unit can generate an output signal that is connected to the ICON unit. The ICON unit then logically ORs all of the external trigger output signals and drives them to a single output pin on the device. Users can enable each ILA core component to drive each respective external trigger output signal to the ICON core component without resynthesizing the design.

Capture Modes

Each ILA core can capture data independently from all other ILA cores in the design. Also, the ILA core can capture data using one of two capture modes: *one shot* and *on trigger*.

The one shot capture mode uses a single trigger event (such as a Boolean or macro combination of the individual trigger match unit events) to collect enough data to fill the sample buffer (up to 4096 samples). The trigger position can be set to the beginning of the sample buffer (trigger first, then collect), the end of the sample buffer (collect until the trigger event), or anywhere in between.

The on trigger capture mode uses multiple trigger events to perform repetitive measurements on the design under test. Each trigger event can cause a capture of 1 to 16 data samples. These repetitive measurements can continue until the captured data fills the sample buffer.

ILA and ICON Core Resource Usage

Tables 1-4, 1-5, 1-6, and 1-7 show the ILA core and ICON core resource usage.

Table 1-4: **ICON and ILA Core CLB Resource Usage in Virtex-II / Virtex-II Pro Devices**

Trigger / Data Width	LUTs	Flip-Flops
8	108	168
16	120	200
32	144	264
64	194	494

Note: A single ILA core with trigger same as data, a single basic match unit, and 512 data samples was used in this example.

Table 1-5: **ICON and ILA Core Block RAM Resource Usage in Virtex-II / Virtex-II Pro Devices**

Trigger / Data Width	Data Samples					
	512	1024	2048	4096	8192	16384
8	1	1	1	2	2	8
16	1	1	2	4	8	16
32	1	2	4	8	16	32
64	2	4	8	16	32	64

Table 1-6: **ICON and ILA Core CLB Resource Usage in Virtex / Virtex-E / Spartan-II / Spartan-IIE Devices**

Trigger / Data Width	LUTs	Flip-Flops
8	106	197
16	118	229
32	145	295
64	194	425

Note: A single ILA core with trigger same as data, a single basic match unit, and 256 data samples was used in this example.

Table 1-7: **ICON and ILA Core Block RAM Resource Usage in Virtex / Virtex-E / Spartan-II Devices**

Trigger / Data Width	Data Samples				
	256	512	1024	2048	4096
8	1	1	2	4	8
16	1	2	4	8	16
32	2	4	8	16	32
64	4	8	16	32	64

Synthesis/Insertion Requirements

Users can modify many options in the ILA and ICON cores without resynthesizing (in the case of the Core Generator) or reinserting (in the case of the Core Inserter). However, after changing selectable parameters (such as width of the data port or the depth of the sample buffer), the design must be resynthesized either with new cores, or with the cores reinserted. Table 1-8 describes which design changes require this.

Table 1-8: **Design Parameter Changes Requiring Resynthesis**

Design Parameter Change	Resynthesis or Re-Insertion Required
Change trigger pattern	No
Running and stopping the trigger	No
Enabling the external triggers	No
Changing the trigger signal source	No[1]
Changing the data signal source	No[1]
Changing the ILA clock signal	Yes
Changing the sample buffer depth	Yes

1. The ability to change existing trigger and/or data signal source is supported by the Xilinx ISE Alliance 4.2i (or later) FPGA Editor or Xilinx ISE Foundation 4.2i (or later) FPGA Editor.

System Requirements

Software Tools Requirements

The ChipScope Core Generator and the ChipScope Core Inserter require that Xilinx ISE Alliance 4.2i (or later) or Xilinx ISE Foundation 4.2i (or later) implementation tools be installed on your system.

Note: The Xilinx WebPACK™ implementation tools are not supported by the ChipScope 4.2i software.

Communications Requirements

The ChipScope Analyzer tool can use either the MultiLINX, Parallel Cable III, or Parallel Cable IV cables to communicate with the target devices in the Boundary Scan chain of the board-under-test.

Table 1-9: **Download Cable Description**

Download Cable	Features
MultiLINX	• Uses the USB port found on newer PCs • Downloads at speeds up to 12 Mb/s throughput • Supports communication through the PC's RS-232 serial port at speeds up to 57.6 kb/s • Contains an adjustable voltage interface that enables it to communicate with systems and I/Os operating at 5V, 3.3V, or 2.5V
Parallel Cable III or IV	• Uses the parallel port (i.e., printer port) to communicate with the Boundary Scan chain of the board-under-test

Note: The MultiLINX, Parallel Cable III, and Parallel Cable IV cables are available from Silicon Xpresso™ Cafe (from www.xilinx.com choose **`Buy Online → Programming Cables`**).

Board Requirements

For the ChipScope Analyzer and download cable to work properly with the board-under-test, the following board-level requirements must be met:

- One or more Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Spartan™-II, Spartan-IIE target devices must be connected to a JTAG header that contains the TDI, TMS, TCK, and TDO pins
- If another device would normally drive the TDI, TMS, or TDI pins of the JTAG chain containing the target device(s), then jumpers on these signals are required to disable these sources, preventing contention with the download cable
- If using the MultiLINX download cable, V_{CC} (2.5-5.0V) and GND headers must be available for powering the MultiLINX cable
- If using the Parallel Cable III download cable, V_{CC} (2.5-3.3V) and GND headers must be available for powering the Parallel Cable III cable
- If using the Parallel Cable III download cable, VREF (2.5-3.V) and GND headers must be available for connecting to the Parallel Cable IV cable

Host System Requirements for Windows NT/98/2000

The ChipScope Analyzer and ChipScope Core Generator tools run on PC systems that meet the requirements outlined in Table 1-10.

Table 1-10: **Analyzer and Core Generator PC System Requirements**

OS	Memory	Supported Ports		Environment
		For MultiLINX:	For Parallel Cable III or IV:	
Windows NT 4.0	64 MB	RS-232	Parallel	Java Run-time Environment version 1.3.1 (automatically included in ChipScope 4.2i software installation)
Windows 98	32 MB	RS-232 or USB	Parallel	
Windows 98 SE	32 MB	RS-232 or USB	Parallel	
Windows 2000	32 MB	RS-232 or USB	Parallel	

Notes:

1. To use the MultiLINX USB interface under Windows 98 Second Edition or Windows 2000, the correct driver must be used. The updated driver is included in all versions of Xilinx software beginning with 3.1i Service Pack 1.

In addition to the requirements outlined in Table 1-10, the ChipScope Core Inserter requires that Xilinx ISE Alliance 4.2i (or later) or Xilinx ISE Foundation 4.2i (or later) implementation tools reside on the system.

Note: The Core Inserter is not compatible with the WebPack implementation tools.

Host System Requirements for Solaris 2.6, 2.7, and 2.8

The ChipScope Core Inserter and ChipScope Core Generator tools run on SPARC workstations that meet these minimum requirements:

- Solaris 2.6, 2.7, or 2.8 operating system capable of running Sun Java 1.3 (See the **`README.sparc`** file in the ChipScope Core Inserter installation directory for more details.)
- Xilinx ISE Alliance 4.2i (or later) or Xilinx ISE Foundation 4.2i (or later) implementation tools
- LD_LIBRARY_PATH environment variable must point to the Xilinx shared libraries
- XILINX environment variable must point to the Xilinx tool installation path
- Path variable must include $XILINX/bin/sol directory

  Note: The Xilinx WebPACK implementation tools are not supported by the ChipScope 4.2i software.

ChipScope ILA Software Installation

Installing ChipScope ILA Software for Windows NT/98/2000

After downloading the ChipScope Tools in the form of a self-extracting executable file (i.e., **ChipScope_*V_Ri*_pc.exe**):

1. Choose **Start** → **Run**.
2. Browse for **ChipScope_*V_Ri*_pc.exe**.
3. Choose **Run**.
4. Follow the install wizard instructions.

Notes:

1. The ChipScope Analyzer can be installed separately from the ChipScope Core Inserter and ChipScope Core Generator tools by performing a "Custom" installation.
2. The Parallel Cable III / IV driver can be installed by performing a "Custom" installation.

Installing the Java Run-time Environment for Windows NT/98/2000

The Java Run-time Environment (JRE) version 1.3.1 used by the ChipScope 4.2i tools is automatically included under the ChipScope 4.2i installation directory.

Installing MultiLINX USB Driver for Windows 98/2000

If you need to install the MultiLINX cable under Windows 98 or Windows 2000 for USB:

1. Make sure that the PWR and GND wires of the MultiLINX cable are connected to power and ground sources, respectively.
2. Plug the cable into the USB port of the host computer. (An installation dialog box opens.)
3. Select **Have Disk**.
4. **Browse** the ChipScope Tools installation for the **mltlnx.inf** file. This is typically installed in folder **C:\Program Files\Xilinx\ChipScope\data**.
5. Click **OK** and follow the installation wizard instructions.

Installing ChipScope Software for Solaris 2.6, 2.7, and 2.8

The ChipScope Core Generator and ChipScope Core Inserter tools for Solaris 2.6, 2.7, and 2.8:

- Java Run-time Environment 1.3.1 (included in download file)
- Solaris operating system patches (for more details, refer to the **README.sparc** file found in the installation archive)

After downloading the tools in the form of a compressed tape archive file (e.g., **ChipScope_V_Ri_sol.tgz** or **ChipScope_V_Ri_sol.zip**):

1. Make sure that the "gzip" and "tar" programs are in your executable path.
2. Change directory to the directory that will contain the ChipScope files.
3. Un-zip and un-tar the **ChipScope_*V_Ri*_sol.tgz** file using the following command:
   ```
   gzip -cd ChipScope_V_Ri_sol.tgz | tar xvf -
   ```
   This will create a chipscope directory under the current working directory.
4. Set up the CHIPSCOPE environment variable to point to the chipscope installation:

   For csh:

```
setenv CHIPSCOPE /path_to_chipscope_parent/chipscope
```

For sh:

```
>  set CHIPSCOPE=/path_to_chipscope_parent/chipscope
>  export CHIPSCOPE
```

5. Run the ChipScope Core Inserter and ChipScope Core Generator tools:

   ChipScope Core Inserter:

   ```
   $CHIPSCOPE/inserter/bin/inserter.sh
   ```

   ChipScope Core Generator:

   ```
   $CHIPSCOPE/inserter/bin/gengui.sh
   ```

XILINX®

Chapter 2

Using the ChipScope Core Generator

Core Generator Overview

The ChipScope Core Generator tool offers is a graphical user interface to generate the Integrated Controller (ICON) Core and the Integrated Logic Analyzer (ILA) Core. Once the cores are generated, users can use the instantiation templates (that are provided) to quickly and easily insert the cores into their VHDL or Verilog design. After completing the instantiation and running synthesis, users implement the design using the Xilinx implementation tools.

Generating an ICON Core

The Core Generator gives users the ability to define and generate a customized Integrated Controller (ICON) unit to use with one or more Integrated Logic Analyzer (ILA) units in VHDL and Verilog designs. Users can customize control ports (the number of ILA cores to be connected to the ICON Core) and control whether to use USER2 Boundary Scan port signals.

After the Core Generator validates the user-defined parameters, it generates an EDIF netlist (*.edn), a netlist constraint file (*.ncf), and example code in VHDL and Verilog specific to the synthesis tool used. Users can easily generate the netlist and code examples for use in normal Virtex™, Virtex-E, Virtex-II, Virtex-Pro™, Spartan™-II, and Spartan-IIE design flows.

The first screen in the Core Generator offers the choice to generate either an ICON core or an ILA core. Select **`ICON (Integrated Controller)`**Core, and click **`Next`**.

Choosing the File Destination

The destination for the ICON EDIF file (**`icon.edn`**) is displayed in the **`Output Netlist`** field. The default directory is the Core Generator install path. To change it, the user can either type a new path in the field, or choose **`Browse`** to navigate to a new destination.

Selecting the Target Device Family

The default target device family is "Virtex / Virtex-E / Spartan-II / Spartan-IIE". ICON cores generated for this family will work for all Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Spartan-II and Spartan-IIE devices. Cores generated for the "Virtex-II / Virtex-II Pro" family will only work for Virtex-II and Virtex-II Pro devices.

Entering the Number of Control Ports

The ICON core can communicate with up to 15 ILA core units at any given time. However, no ILA core unit can share its control port with any other ILA unit. Therefore, the ICON core needs up to 15 distinct control ports to handle this requirement. Users can select the number of control ports from the **Number of Control Ports** pull-down list.

Enabling the External Triggers

Users can configure the ICON core to implement an external trigger input and an external trigger output (i.e. two separate pins) to trigger external test equipment. To enable instantiation of the external input and output trigger pins, select the appropriate **Enable External Trigger Input** and **Enable External Trigger Output** check box. If external trigger input or output pins are enabled, then the pin locations that are entered in the corresponding text boxes are written to the NCF constraint file of the ICON core. If the design does not require external triggers, or if there are no usable pins for triggers, do not check these boxes.

Disabling JTAG Clock BUFG Insertion

Disabling the JTAG clock BUFG insertion causes the implementation tools to route the JTAG clock using normal routing resources instead of global clock routing resources. By default, this clock is placed on a global clock resource (BUFG). To disable this BUFG insertion, check select the Disable JTAG Clock BUFG Insertion check box. This should only be done if global resources are very scarce; placing the JTAG clock on regular routing, even high-speed backbone routing, introduces skew. Make sure the design is adequately constrained to minimize this skew.

Including Boundary Scan Ports

The BSCAN_VIRTEX primitive has two sets of ports: USER1 and USER2. These provide an interface to the Boundary Scan TAP controller of the Virtex or Virtex-E device. Since the ICON core uses only the USER1 port for communication purposes, the USER2 port signals are available. To use the USER2 interface to the BSCAN_VIRTEX primitive, select the **Include Boundary Scan Ports** check box.

> Note: The Boundary Scan ports should be included *only* if the design needs them. If they are included and not used, some synthesis tools do not connect the ICON core properly, causing errors during the synthesis and implementation stages of development.

Selecting the Instantiation Template

After selecting the parameters for the ICON core, you can construct an instantiation template. Click **Next** to view the Sample Code Generation Options, then select which synthesis tool and language to use. The synthesis tools supported are:

- Exemplar LeonardoSpectrum™
- Synopsys FPGA Compiler™
- Synopsys FPGA Compiler II™
- Synopsys FPGA *Express*™
- Synplicity Synplify®
- XST (Xilinx Synthesis Technology)

Specifically tailored attributes and options are embedded in the HDL instantiation template for the various synthesis tools. To generate the ICON core without any example files, deselect the **Generate Example Files** check box.

Generating the Core

After entering the ICON core parameters, click **Generate Core** to create the EDIF netlist, NCF constraint file, and applicable code examples. A message window opens, the progress information appears, and the CORE GENERATION COMPLETE message signals the end of the process. The user can then either go back and respecify different options or **Start Over**.

Using the ICON Core

To instantiate the example ICON core HDL files into your design, use the following guidelines to connect the ICON core port signals to various signals in your design:

- Connect one of the ICON core's unused CONTROL* port signals to a control port of only one ILA core instance in the design
- Do not leave any unused CONTROL* ports of the ICON core unconnected as this will cause the implementation tools to report an error. Instead, use an ICON core with the same number of CONTROL* ports as you have ILA cores

Generating an ILA Core

The ChipScope Core Generator allows users to define and generate a customized ILA capture core to use with VHDL and Verilog designs. Users can customize the maximum number of data sample words stored by the ILA core, the width of the data sample words, and the width of the trigger word (if different from the data word).

After the Core Generator validates the user-defined parameters, it generates an EDIF netlist (***.edn**), a netlist constraint file (***.ncf**), and example code specific to the synthesis tool used. Users can easily generate the netlist and code examples for use in normal Virtex, Virtex-E, Virtex-II, Virtex-Pro, and Spartan-II, and Spartan-IIE design flows.

The first screen in the Core Generator offers the choice to generate either an ICON or ILA core. Select ILA (Integrated Logic Analyzer), and click **Next**.

Choosing the File Destination

The destination for the ILA EDIF (ila.edn) is displayed in the **Output Netlist** field. The default directory is the Core Generator install path. To change it, the user can either type a new path in the field, or choose **Browse** to navigate to a new destination.

The user can select from three types of names: a long name in which the options are specified in the component name; a short name (**ila.edn**); or a custom name for the netlist (the component name will be changed accordingly). Either a long name or a custom name should be chosen if multiple ILA cores with different parameters are used in the design.

The long filename indicates the various parameters specified for the ILA core. Table 2-1 shows the meaning of the abbreviations in a long filename.

Table 2-1: **A Long Filename Abbreviations**

Abbreviation	Meaning
dd*x*	Data Depth of size *x*
dw*x*	Data Width of size *x*
tw*x*	Trigger Width of size *x*
t_eq_d	Trigger Equals Data
e*x* or b*x*	Extended or Basic Matching with *x* Units

Selecting the Target Device Family

The target FPGA device family is displayed in the Device Family field. The structure of the ILA core is optimized for the selected device family. Use the pull-down selection to change the device family to the desired architecture. Note that the default target device family is "Virtex / Virtex-E / Spartan-II". ILA cores generated for this family will work for all Virtex, Virtex-E, Virtex-II and Spartan-II devices. Cores generated for the Virtex-II family will only work for Virtex-II devices.

The **`Use SRL16's`** check box is used to select whether or not the ILA will be generated using SRL16 and SRL16E components. If the check box is not selected, the SRL16 components are replaced with flip-flops and multiplexers. Note that the default target device family is "Virtex / Virtex-E / Spartan-II / Spartan-IIE". ILA cores generated for this family will work for all Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Spartan-II and Spartan-IIE devices. Cores generated for the "Virtex-II / Virtex-II Pro" family will only work for Virtex-II and Virtex-II Pro devices.

Selecting the Clock Edge

The ILA unit can use either edge of the CLK signal to trigger and store data. This option is used to select either the rising or falling edge of the CLK signal as the clock source for the ILA core.

Selecting the Trigger Type

To generate the first part of the ILA core, select one of the following trigger types:

- **Trigger separate from data**

  The trigger word is completely independent of the data word.

- **Trigger same as data**
  - The trigger and data words are identical. This mode is very common in most logic analyzers, since users can trigger on any bit in the data word being collected.
  - If this selection is made, then the TRIG input port of the ILA core can be left unconnected.
  - The DATA input port is used as both data and trigger input to the ILA core. This mode also conserves CLB and routing resources in the ILA core, but limits the data sample word width to the maximum trigger width of 64 bits.

Selecting the Trigger Match Unit Type

An ILA core trigger unit comprises one or more match units that contribute to the overall trigger condition by looking for a specific pattern on the trigger input. The types of patterns and their occurrence over time that are looked for depend on the type of the match unit. The ChipScope core generator handles *basic* and *extended* trigger match units.

The basic trigger match unit:

- Finds only one occurrence of an exact match of a trigger value or edge
- Can be used in conjunction with other trigger match units to build the trigger condition Boolean equation (using AND/OR)

The extended trigger match unit:

- Finds **one or more** occurrences of an exact match of a trigger value or edge
- Finds **one or more** occurrences of a range of trigger values
- Detects contiguous (pulse width) or non-contiguous (event count) trigger match conditions over a number of clock cycles
- Can be used in conjunction with other trigger match units to build the trigger condition Boolean equation (using AND/OR)
- Can be used in conjunction with other trigger match units to build the trigger condition macro equation (using IF/THEN)

For the ILA core that is being generated, the Trigger Match Unit Type is selected for all match units at the same time.

Selecting the Number of Trigger Match Units

Selecting one match unit conserves resources while still allowing some flexibility in triggering. The number of Trigger Match Units can be set to either one or two. Selecting two trigger match units allows a more flexible trigger condition equation to be a combination of both match units.

Selecting the Data Depth

The maximum number of data sample words that the ILA core can store is called the *data depth*. The data depth determines the number of data width bits contributed by each block RAM unit used by the ILA unit.

For the Virtex, Virtex-E, Spartan-II and Spartan_IIE device families, you can set the data depth to one of five values as shown in Table 2-2.:

Table 2-2: **Maximum Data Widths (Virtex / Virtex-E / Spartan-II / Spartan-IIE Devices)**

	Depth 256	Depth 512	Depth 1024	Depth 2048	Depth 4096
1 block RAM	16	8	4	2	1
2 block RAMs	32	16	8	4	2
4 block RAMs	64	32	16	8	4
8 block RAMs	128	64	32	16	8
16 block RAMs	256	128	64	32	16
32 block RAMs	--	256	128	64	32
64 block RAMs	--	--	256	128	64
128 block RAMs	--	--	--	256	128
256 block RAMs	--	--	--	--	256

For the Virtex-II and Virtex-II Pro device families, you can set the data depth to one of six values as shown in Table 2-3.:

Table 2-3: **Maximum Data Widths (Virtex-II / Virtex-II Pro Devices)**

	Depth 512	Depth 1024	Depth 2048	Depth 4096	Depth 8192	Depth 16384
1 block RAM	32	16	8	4	2	1
2 block RAMs	64	32	16	8	4	2
4 block RAMs	128	64	32	16	8	4
8 block RAMs	256	128	64	32	16	8
16 block RAMs	--	256	128	64	32	16
32 block RAMs	--	--	256	128	64	32
64 block RAMs	--	--	--	256	128	64
128 block RAMs	--	--	--	--	256	128
144 block RAMs	--	--	--	--	--	144

Entering the Data Width

The width of each data sample word stored by the ILA core is called the *data width*. If the data and trigger words are independent from each other, then the maximum allowable data width depends on the target device type and data depth. However, if the data and trigger words are the same, then the data/trigger width must be any even number in the ranging from 2 to 64.

Selecting the Trigger Width

The width of the trigger word used by the ILA core is called the *trigger width*. If the data and trigger words are independent from each other, then the trigger width can be any even integer in the range of 2 to 64. However, if the data and trigger words are the same, then both the data and trigger widths must be set to any even integer value in the range of 2 to 64.

Selecting the Instantiation Template

After selecting the parameters for the ILA core, you can construct an instantiation template. Click **`Next`** to view the Sample Code Generation Options, then select which synthesis tool and language to use. Supported synthesis tools are:

- Exemplar LeonardoSpectrum™
- Synopsys FPGA Compiler™
- Synopsys FPGA Compiler II™
- Synopsys FPGA *Express*™
- Synplicity Synplify®
- XST (Xilinx Synthesis Technology)

Specifically tailored attributes and options are embedded in the HDL instantiation template for the various synthesis tools. To generate the ILA core without any example files, deselect the **`Generate Example Files`** check box.

Generating the Core

After entering the ILA core parameters, click **Generate Core** to create the netlist and applicable code examples. A message window opens, the progress information appears, and the CORE GENERATION COMPLETE message signals the end of the process. The user can select to either go back and specify different options or **Start Over**.

Using the ILA Core

To instantiate the example ILA core HDL files into your design, use the following guidelines to connect the ILA core port signals to various signals in your design:

- Connect the ILA core's CONTROL port signal to an **unused** control port of the ICON core instance in the design
- Connect all unused bits of the ILA core's data and trigger port signals to "1". This prevents the mapper from removing the unused trigger and/or data signals and also avoids any DRC errors during the implementation process
- If **Trigger Same As Data** is selected, connect the data/trigger signal(s) to the ILA core's DATA port signal. The ILA core's TRIG port signal is disconnected in this case. In the source code, this port can be connected to the same bus as the DATA port or to all "1"s
- Make sure the data and trigger source signals are synchronous to the ILA clock signal

Chapter 3

Using the ChipScope Core Inserter

Core Inserter Overview

The ChipScope Core Inserter is a post-synthesis tool used to generate a netlist that includes the user design as well as ICON and ILA cores, parameterized accordingly. The Core Inserter gives users the flexibility to quickly and easily use the ILA functionality to an already synchronized design, and without any HDL instantiation.

ChipScope Core Inserter Menu Features

Working with Projects

Projects saved in the Core Inserter hold all relevant information about source files, destination files, core parameters, and core settings. This allows the user to store and retrieve information about core insertion between sessions.

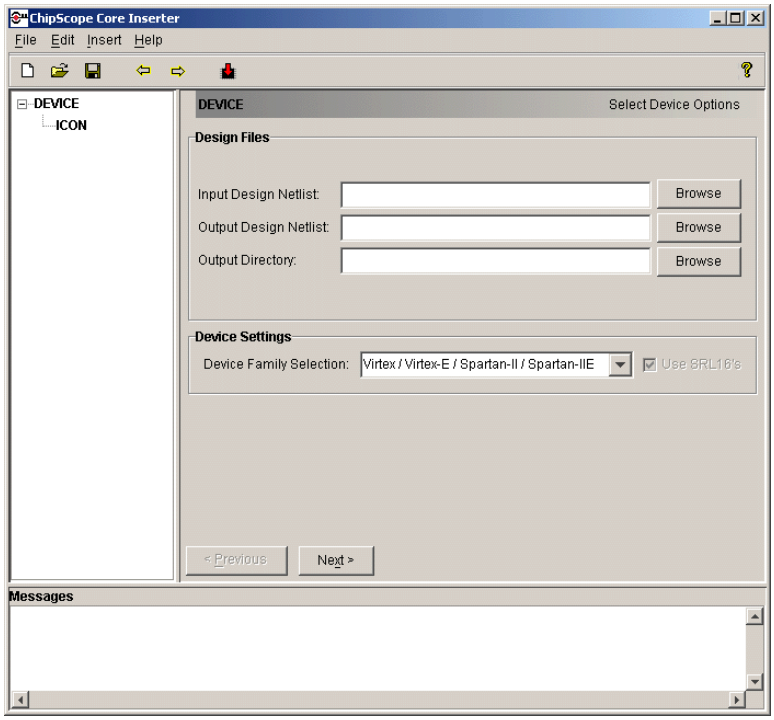

Figure 3-1: **Blank Core Inserter Project**

When the ChipScope Core Inserter is first opened, all the relevant fields are completely blank. Using the command **`File`** → **`New`** also results in this condition (Figure 3-1).

Opening an Existing Project

To open an existing project, select it from the list of recently opened projects, or select **File** → **Open Project**, and **Browse** to the project location. When the desired project is located, double-click on it, or click **Open**.

Saving Projects

If a project has changed during the course of a session, the user will be prompted to save the project upon exiting the Core Inserter. A project can also be saved by selecting **File** → **Save**. To rename the current project or save it to another filename, select **File** → **Save As**, type in the new name, and click **Save**.

Refreshing the Netlist

The Core Inserter automatically reloads the design netlist if it detects that the netlist has changed since the last time it was loaded. However, the Core Inserter can be forced to refresh the netlist by selecting **File** → **Refresh Netlist**.

Inserting and Removing ILA Units

New ILA units can be inserted into the project by selecting **Edit** → **New ILA Unit**. An ILA unit can be removed by selecting **Edit** → **Remove Unit** after choosing which ILA unit to delete.

Setting Preferences

Several ChipScope Core Inserter project settings can be customized by selecting **Edit** → **Preferences**. The preference settings are organized into three categories: **Tools**, **ISE Integration**, and **Miscellaneous**. Refer to the **Managing Project Preferences**, page 10 section of this chapter for more information about setting these preferences.

Inserting the Cores

ICON and ILA cores are inserted when the flow is completed, or by selecting **Insert** → **Insert Core**. If all channels of all the ILA cores are not connected to valid signals, an error message results.

Exiting the Core Inserter

To exit the ChipScope Core Inserter, select **File** → **Exit**.

Specifying Input and Output Files

The ChipScope Core Inserter works in a step-by-step process. The first screen you see (Figure 3-1) shows what needs to be specified first: the input design netlist. Click **Browse** to navigate to the directory where the netlist resides. The Core Inserter will not overwrite the netlist specified. Instead, a new netlist will be created with the _ila extension appended by default. The Output Design Netlist and Output Directory fields are automatically filled

in, however, you can modify them as needed. Figure 3-2 shows a project with input and output files specified.

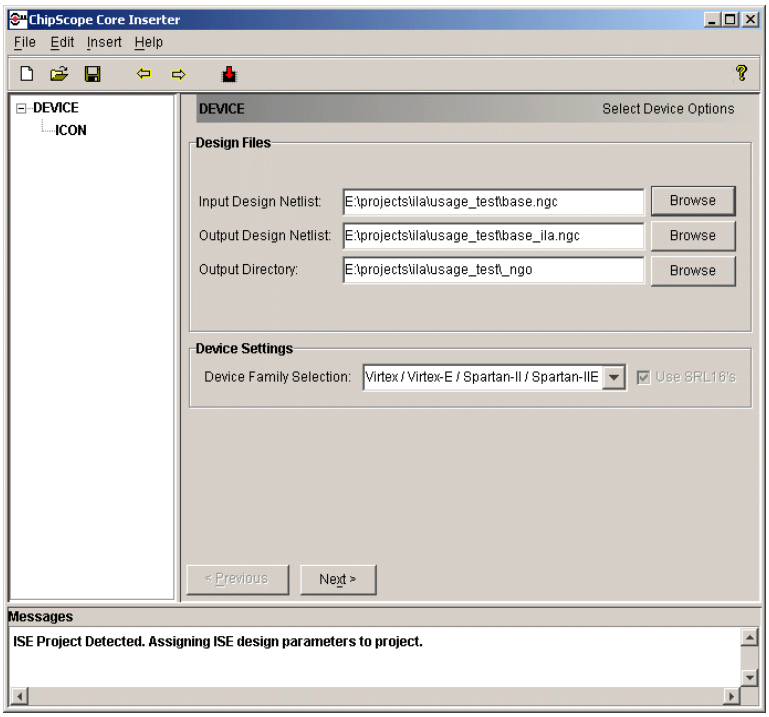

Figure 3-2: **Core Inserter Project with Files Specified**

Project Level Parameters

Two project level parameters must be specified for every project. Due to the increased depth of Virtex-II device block RAM, different cores can be generated to take advantage of these deeper RAMs. Choose either Virtex/Virtex-E/Spartan-II or Virtex-II/Virtex-II Pro, depending on the device targeted. Also, it is possible to generate cores that do not use the SRL16 (Shift Register LUT) components; to do so, uncheck the **`Use SRL16's`** check box.

Selecting the Target Device Family

The target FPGA device family is displayed in the Device Family field. The structure of the ICON and ILA cores are optimized for the selected device family. Use the pull-down selection to change the device family to the desired architecture.The default target device family is "Virtex / Virtex-E / Spartan-II / Spartan-IIE". Cores generated for this family will work for all Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Spartan-II and Spartan-IIE devices. Cores generated for the Virtex-II / Virtex-II Pro family will only work for Virtex-II and Virtex-II Pro devices.

The **`Use SRL16's`** check box is used to select whether or not the cores will be generated using SRL16 and SRL16E components. If the check box is not selected, the SRL16 components are replaced with flip-flops and multiplexers. The replacement of SRL16 components with flop-flops and multiplexers will affect the size and performance of the generated cores. The **`Use SRL16's`** check box is checked by default in order to generate cores that use the optimized SRL16 technology.

When this step is completed, click **`Next`**.

Choosing ICON Options

The first options that need to be specified are for the ICON core. The ICON core is the controller core that connects all ILA units to the JTAG boundary scan chain. The ICON core has the options parameters shown in Figure 3-3.

Enable External Trigger Input

Enabling the external trigger input causes a PAD to be added to the design; the pin location specified in Figure 3-3 is added in the NCF file that is automatically generated for the ICON core. This signal is a logical OR'd condition of all the trigger states of all ILA cores in the design.

Enable External Trigger Output

Enabling the external trigger output causes a PAD to be added to the design; the pin location specified in Figure 3-3 is added in the NCF file that is automatically generated for the ICON core. This signal can be used in the ChipScope Analyzer to trigger the ILA cores.

Disable JTAG Clock BUFG Insertion

Disabling the JTAG clock BUFG insertion causes the implementation tools to route the JTAG clock using normal routing resources instead of global clock routing resources. By default, this clock is placed on a global clock resource (BUFG). To disable this BUFG insertion, check select the **Disable JTAG Clock BUFG Insertion** check box. This should only be done if global resources are very scarce; placing the JTAG clock on regular routing, even high-speed backbone routing, introduces skew. Make sure the design is adequately constrained to minimize this skew.

After selecting the desired ICON options (Figure 3-3), click **Next**.

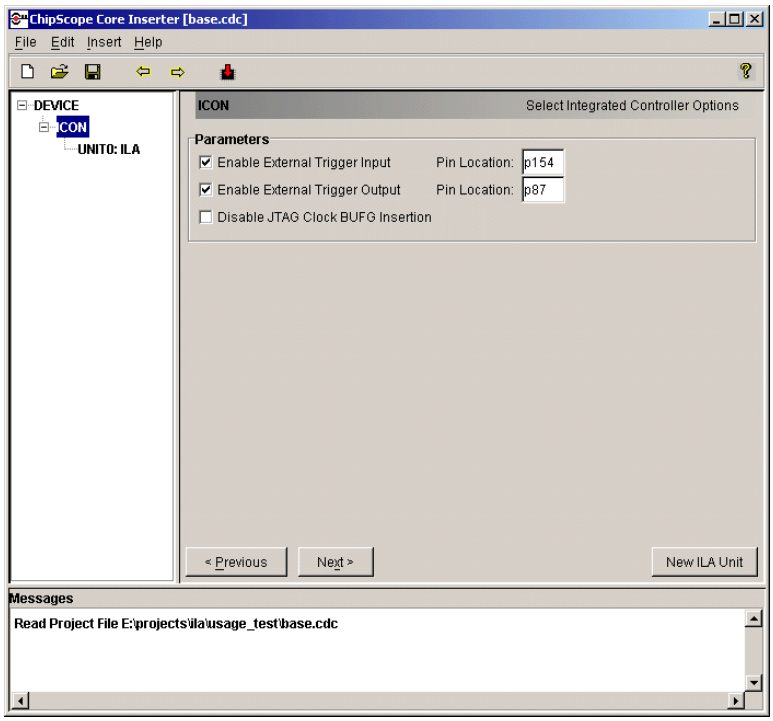

Figure 3-3: **ICON Options**

Choosing ILA Parameters and Options

Notice in Figure 3-3 that a new ILA unit has been created in the device hierarchy on the left. The next step is to set up the ILA unit. Figure 3-4 shows a sample of the ILA options and parameters.

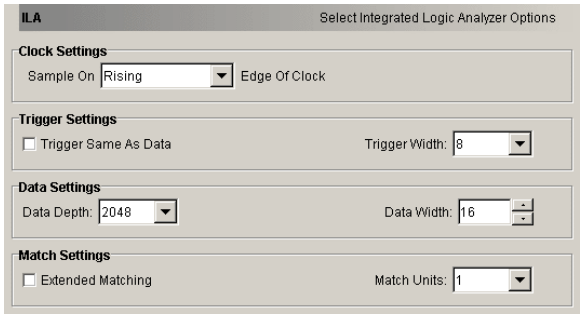

Figure 3-4: **ILA Options and Parameters**

Clock Settings

The ILA unit can use either edge of the CLK signal to trigger and store data. This option is used to select either the rising or falling edge of the CLK signal as the clock source for the ILA core.

Trigger Settings

Trigger Same as Data

Use this option when the signals that you want to trigger on are exactly the same signals that you want captured. This option also conserves CLB and routing resources in the ILA core, but limits the data sample word width to the maximum trigger width of 64 bits. This is the common mode with most logic analyzers. Notice that when this option is checked, the Data Width field disappears and Trigger Width is renamed to Trigger/Data Width.

Trigger Width

This specifies the width of the trigger bus, or the width of the Trigger/Data bus when trigger is the same as data. Valid numbers are even integers from 2 to 64.

Data Settings

Data Depth

The maximum number of data sample words that the ILA core can store is called the *data depth*. The data depth determines the number of data width bits contributed by each block RAM unit used by the ILA unit.

For the Virtex, Virtex-E, Spartan-II, and Spartan-IIE device families, you can set the data depth to one of five values shown in Table 3-1:

Table 3-1: **Maximum Data Widths (Virtex / Virtex-E / Spartan-II / Spartan-IIE Devices)**

	Depth 256	Depth 512	Depth 1024	Depth 2048	Depth 4096
1 block RAM	16	8	4	2	1
2 block RAMs	32	16	8	4	2
4 block RAMs	64	32	16	8	4
8 block RAMs	128	64	32	16	8
16 block RAMs	256	128	64	32	16
32 block RAMs	--	256	128	64	32
64 block RAMs	--	--	256	128	64
128 block RAMs	--	--	--	256	128
256 block RAMs	--	--	--	--	256

For the Virtex-II and Virtex-II Pro device families, you can set the data depth to one of six values shown in Table 3-2:

Table 3-2: **Maximum Data Widths (Virtex-II and Virtex-II Pro Devices)**

	Depth 512	Depth 1024	Depth 2048	Depth 4096	Depth 8192	Depth 16384
1 block RAM	32	16	8	4	2	1
2 block RAMs	64	32	16	8	4	2
4 block RAMs	128	64	32	16	8	4
8 block RAMs	256	128	64	32	16	8
16 block RAMs	--	256	128	64	32	16
32 block RAMs	--	--	256	128	64	32
64 block RAMs	--	--	--	256	128	64
128 block RAMs	--	--	--	--	256	128
144 block RAMs	--	--	--	--	--	144

Data Width (if necessary)

The data width field allows you to specify the width of each data sample stored by the ILA core. If the data and trigger words are independent from each other, the maximum allowable data width depends on the target device type and data depth, with a maximum of 256. However, if trigger = data, the data/trigger width must be an even number between 2 and 64.

Match Settings

Extended Matching

Check the Extended Matching check box if you want to enable a complete set of match options. The match options include:

- Finding one or more occurrences of an exact match of a trigger value or edge
- Finding one or more occurrences of a range of trigger values
- Detecting contiguous (pulse width) or non-contiguous (event count) trigger match conditions over a number of clock cycles
- Enabling usage in conjunction with other trigger match units to build the trigger condition boolean equation (using AND/OR)
- Enabling usage in conjunction with other trigger match units to build the trigger condition macro equation (using IF/THEN)

Basic Matching

If Extended Matching is not checked, a subset of the above features are available for all match units, saving CLB usage of the ILA core. These options include:

- Finding only one occurrence of an exact match of a trigger value or edge
- Enabling usage in conjunction with other trigger match units to build trigger condition boolean equation (using AND/OR)

Match Units

The number of match units can be set to one or two. Selecting two units allows a more flexible trigger condition equation to be a combination of both match units. Selecting one match unit conserves resources while allowing some flexibility in triggering.

Choosing Net Connections for ILA Signals

The *Net Connections* panel (see Figure 3-5) under the *Match Settings* allows the user to choose the signals to connect to the ILA core. If trigger is separate from data, then Clock, Trigger, and Data must be specified. When trigger equals data, only Clock and Trigger/Data must be specified.

Double-click on the **`Clock Net`** label or click on the plus sign (+) next to it to expand as shown in Figure 3-5 (only the *Net Connections* panel is shown):

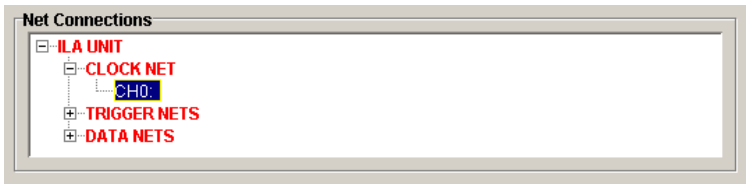

Figure 3-5: **ILA Core Clock Specification**

The **Clock Net** connection can be modified by double-clicking on the **CH0:** label or highlighting it and clicking **Modify Connections**. The *Select Net* dialog box now appears (see Figure 3-6).

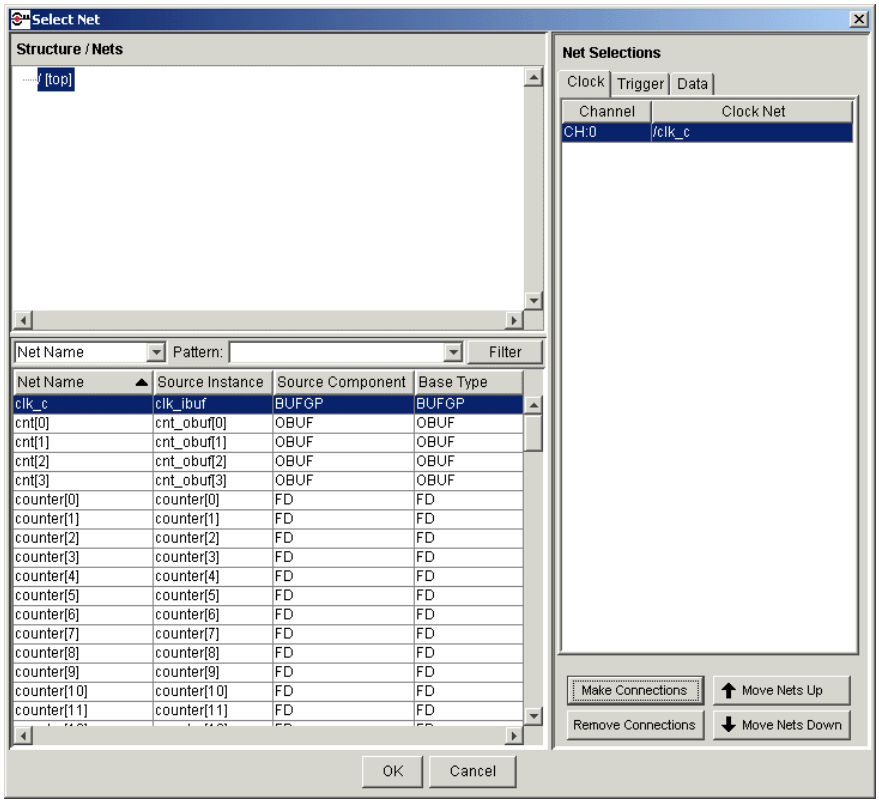

Figure 3-6: **Select Net Dialog Box**

This dialog box provides an easy interface to choose nets to connect to the ILA core. The hierarchical structure of the design can be traversed using the *Structure* panel on the upper left of the *Select Net* dialog box. All the design's nets of the selected structure hierarchy level appear in the table on the lower left panel of the *Select Net* dialog box. The following net information is displayed in this table:

- **Net Name**. The name of the net as it appears in the EDIF netlist. The net name may be different than the corresponding signal name in the HDL source due to renaming and other optimizations during synthesis.
- **Source Instance**. The instance name of the lower-level hierarchical component from which the net at the current level of hierarchy is driven. The source instance does not necessarily describe the originating driver of the net.
- **Source Component**. The type of the component described by the Source Instance.
- **Base Type**. The type of the lowest level driving component of the net. The base type is either a primitive or "black box" component.

All of the net identifiers described above can be filtered for key phrases using the **Pattern** text box and **Filter** button. Also, nets can be sorted in ascending and descending order based on the various net identifiers by selecting the appropriate net identifier button in the column headers of the net selection table.

> **Note**: the net names are sorted in alpha-numeric or "bus element" order whenever possible. Common delimeters such as "[", "(", etc., are used to identify possible bus element nets.

The Clock, Data, and Trigger inputs of the ILA core appear in the tabbed panel at the upper right of the *Select Net* dialog box. Nets that are selected at a given level of hierarchy can be connected to inputs of the ILA core by following these steps:

1. In the lower-left table of the *Select Net* dialog box, select the net(s) that you want to connect to the ILA core.

   Note: You can select multiple nets to connect to an equivalent number of ILA core input connections. Hold down the Shift key and use the left mouse button to select contiguous nets. Use a combination of the **Ctrl** key and left mouse button to select non-contiguous nets.

2. In the upper-right tabbed panel of the *Select Net* dialog box, the desired ILA core input category: Clock, Trigger, Data (or Trigger/Data, if trigger is same as data).

3. In the right-hand table of ILA core inputs, select the channel(s) that you want to connect to the selected net(s).

   Note: You can select multiple ILA core inputs to connect to an equivalent number of nets. Hold down the **Shift** key and use the left mouse button to select contiguous ILA core inputs. Use a combination of the **Ctrl** key and left mouse button to select non-contiguous ILA core inputs.

4. In the lower-right part of the *Select Net* dialog box, click on the **Make Connections** button to make a connection between the selected nets and ILA core inputs.

Use the **Remove Connections** button to remove any existing connections. Use the **Move Up** and **Move Down** buttons to reorder the position of any selected connection. Once the desired net connections have been made, click the **OK** button to return to the main Core Inserter window.

All the nets for Trigger and Data must be chosen in this fashion. After you have chosen all the nets for a given bus, the ILA bus name changes from red to black (see Figure 3-7).

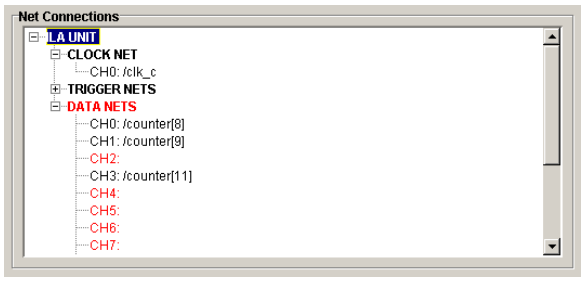

Figure 3-7: **Specifying Data Connections**

After specifying the Clock, Trigger, and Data nets, click **Next**. A dialog box appears asking if you want to proceed with Core Insertion. If Yes, the cores are generated, inserted into the netlist, and an NGO file is created by running the edif2ngd program. Details of this process can be viewed in the **Messages** panel at the bottom of the window. A Core Generation Complete message in the **Messages** panel indicates successful insertion of ChipScope cores.

Adding ILA Units

Each device can support up to 15 ILA units, depending on block RAM availability and ILA unit parameters). To add another ILA unit to the project, select **Edit** → **New ILA Unit**, or going to the ICON Options window by clicking on ICON in the tree on the left panel (see Figure 3-3) and clicking the **New ILA Unit** button. Parameters for the additional ILA units are set up using the same procedure as above.

Inserting Cores into Netlist

The core insertion step can be invoked by selecting the **Insert** → **Insert Core** menu option or by clicking on the **Insert Core** button on the toolbar.

Managing Project Preferences

The preference settings are organized into three categories: **`Tools`**, **`ISE Integration`**, and **`Miscellaneous`**. These preference settings are shown in Figure 3-8, Figure 3-9, and Figure 3-10, respectively.

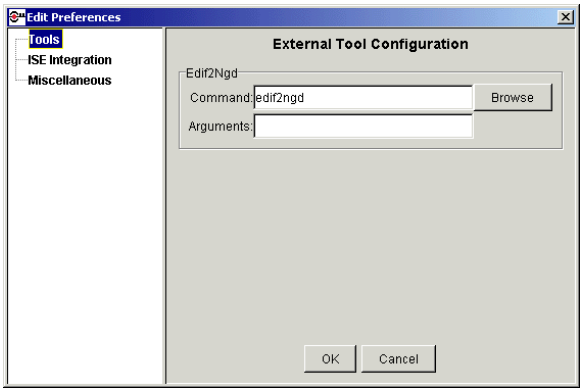

Figure 3-8: **Core Inserter Tools Preference Settings**

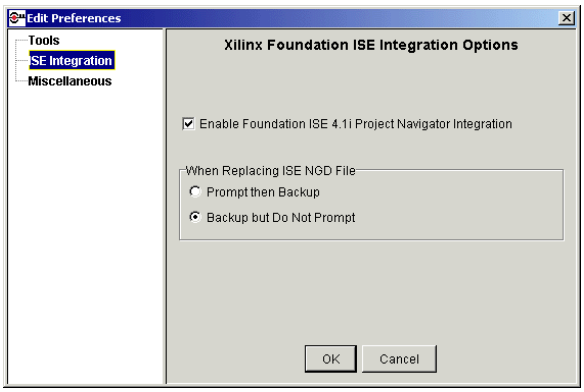

Figure 3-9: **Core Inserter ISE Integration Preference Settings**

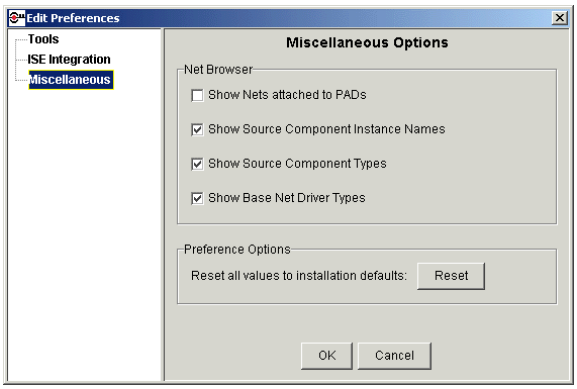

Figure 3-10: **Core Inserter Miscellaneous Preference Settings**

The **`Tools`** section contains settings for the command line arguments used by the Core Inserter to launch the Edif2Ngd tool.

The ISE Integration section contains settings that affect how the Core Inserter integrates with the Xilinx ISE Foundation 4.2i Project Navigator tool. When ISE integration is enabled (by default), the Core Inserter automatically searches through the current working directory for ISE 4.2i temporary netlist directory called **_ngo**. If a valid ISE 4.2i **_ngo** directory is found, the Core Inserter project will be set up automatically to overwrite the intermediate NGD files of the ISE project with those produced by the Core Inserter. The ISE Integration preferences can be set by the user to prompt the user before overwriting any intermediate NGD files.

The **Miscellaneous preferences** section contains other settings that affect how the Core Inserter operates. For instance, the Core Inserter can be set up by the user to display the net names of PAD components in the *Select Net* dialog box. Also, the Core Inserter can be set up by the user to disable the display of source component instance names, source component types, and base net driver types in the Select Net dialog box. The Core Inserter project preferences can also be reset by the user to the installation defaults by clicking on the **Reset** button.

Using Core Inserter 4.2i with Command Line Implementation

Since the Core Inserter generates a file of a different format than synthesis (an NGO file instead of an EDIF or XST netlist), the implementation flow is slightly different. If you use the command line, use the NGO file produced by the Core Inserter instead of the EDIF or XST netlist.

For JTAG configuration:

```
ngdbuild -sd <netlist path> -p <part type> base_ila.ngo
map base_ila
par -ol 5 -w base_ila base_ila
bitgen -w -g UserID:<user ID> -g StartupClk:JTAGClk base_ila base_ila
```

Using Core Inserter 4.2i with Xilinx ISE Foundation 4.2i

The Core Inserter 4.2i with Xilinx ISE Foundation 4.2i implementation tools require some flow manipulation to ensure the correct files are implemented. To insert ChipScope cores via the ChipScope Core Inserter into a design processed by Xilinx ISE Foundation 4.2i, follow these steps:

1. **Synthesize** and **Translate** the design.

   After creating a project and adding the HDL or EDIF/XST source files, **Synthesize** (if required for the selected flow) and **Translate** the design by double-clicking **Translate** under **Implement Design**.

2. Run the ChipScope Core Inserter.

   Launch the Core Inserter from **Start** → **Programs** → **ChipScope 4.2i** → **ChipScope Core Inserter**.

   Note: ChipScope Core Inserter version 4.1i and earlier cannot be used with ISE 4.2i.

3. **Browse** for the top level design to fill in the Input Design Netlist. Verify the following parameters are set under the **Device** tab:

- For Exemplar, Synopsys, and Synplicity designs:
  `Input Design Netlist = <ProjectDir>\<design>.[edf|edn]`
  `Output Design Netlist = <ProjectDir>\_ngo\<design>.ngo`
  `Output Directory = <ProjectDir>\_ngo`
- For XST designs:
  `Input Design Netlist = <ProjectDir>\<design>.ngc`
  `Output Design Netlist = <ProjectDir>\<design>.ngc`
  `Output Directory = <ProjectDir>\_ngo`

Browse for the input ***<design>*.edf**, ***<design>*.edn**, or ***<design>*.ngc** file. The Core Inserter will detect the presence of the ISE project when it finds the **_ngo** directory, and the two output fields will be filled in automatically. This can be done only after the **Translate** step has been run at least once.

4. Complete ICON and ILA core insertion in the ChipScope Core Inserter.

   Define ChipScope Core parameters and connect all signals, then insert the cores. The Core Inserter places new NGO files for the top level, as well as for the ICON and ILA cores, in the **_ngo** directory. Save the project before exiting the Core Inserter.

5. Rerun the **Translate** step.

   When you return to Xilinx ISE Foundation 4.2i, you should see that the green check mark by the **Translate** step has disappeared. Double-click **Translate**, or right-click **Translate** and select "Run". The RUN command will start with the new NGO files.

   Note: DO NOT select "Rerun All," as this will rerun Synthesis and the first portion of **Translate**, which will undo the Core Insertion that was just performed.

6. Continue with **Implementation**.

   Make sure to specify the ChipScope Core Inserter bitstream generation options (JTAG clock when required, enable Readback) when creating the bitstream.

You must follow all of these steps after making any modifications to the HDL source or after deleting any of the implementation data. Simply running through the standard ISE flow will not insert the ChipScope cores into the design.

Chapter 4

Using the ChipScope Analyzer

Analyzer Overview

The ChipScope Analyzer tool interfaces directly to the ILA and ICON cores. Users can configure their device, choose triggers, and view the results of the capture on the fly. The waveforms and triggers can be manipulated in many ways, providing an easy and intuitive interface to determine the functionality of the design.

Analyzer Menu Features

Selecting a Device/ILA Unit

A single target device (i.e., Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Spartan-II, or Spartan-IIE device) can contain up to 15 ILA units. The host communicates through a separate ChipScope Analyzer window for each ILA unit. You can select the current ILA unit only after you connect to the download cable and detect the Boundary Scan chain.

Select **`File`** → **`New ILA Unit`** → **`Unit`** *n* (where *n* is the ILA unit number) to choose a specific ILA unit. Note that the ILA unit number corresponds to the control port the ILA core is connected to in the case of instantiation, or the Unit number in the case of Core Insertion. Figure 4-1 shows how ILA Unit 1 is selected.

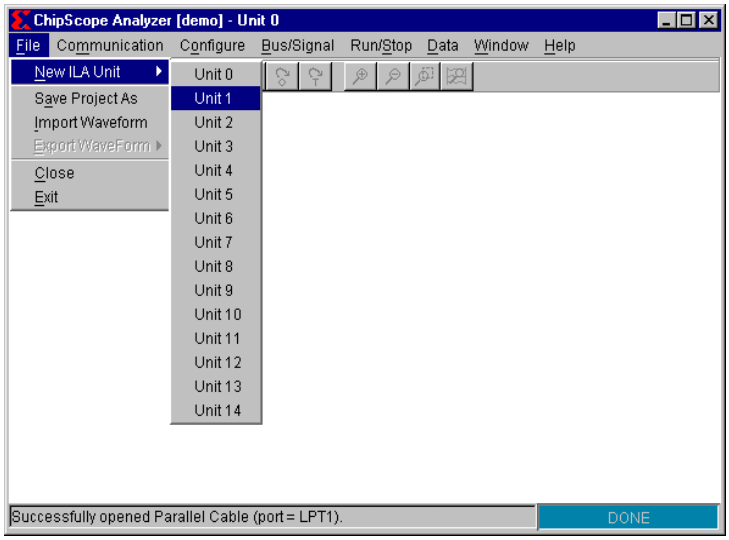

Figure 4-1: **Selecting a New ILA Unit**

If the Boundary Scan chain contains multiple devices that can serve as ILA targets, then select communication to one of the 15 possible ILA units in the target device using **`File`** →

`New ILA Unit` → **`Device`** *m* → **`Device`** *m* **`Unit`** *n* (where *m* is the target device number *n* is the ILA unit number). Note that the ILA unit number corresponds to the control port number of the ICON unit to which the ILA component is connected, or the Unit number in the case of Core Insertion. Figure 4-2 shows how to select ILA Unit 0 in target Device 1.

If the trigger setup toolbar is not present in the current ChipScope window, then selecting a new ILA unit using this method simply refocuses the current window to the new ILA unit. However, if you select a new ILA unit after you have set up the trigger for the current one, a new ChipScope window opens. This feature allows you to view the waveforms of multiple ILA units at the same time.

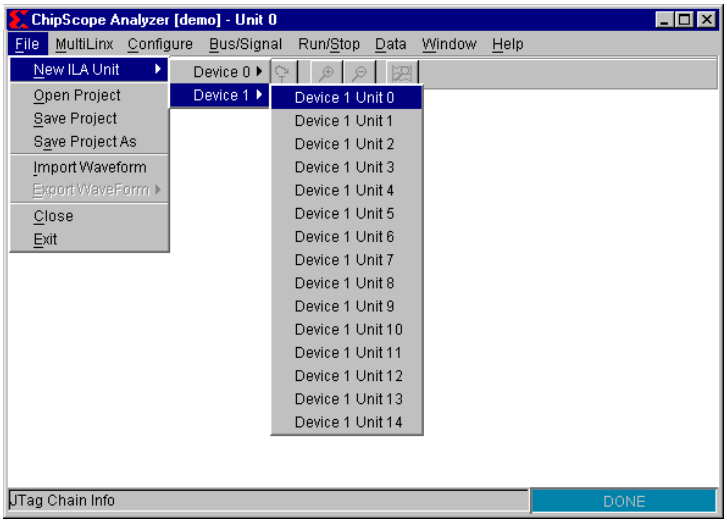

Figure 4-2: **Selecting New ILA Unit in Target Device 1**

New ILA units are displayed in their own window and the Device/Unit for a particular window is displayed in the title bar. All the ILA units may be triggered at the same time by configuring each trigger in turn. There is no current way to combine the waveform display or trigger setup amongst separate ILA units.

Working with Projects

Projects hold important information about the ChipScope program state, such as signal naming, signal ordering, bus configurations, and trigger conditions. They allow you to conveniently store and retrieve this information between Analyzer sessions

When you first run the ChipScope Analyzer tool, the **`Select Project`** dialog box (Figure 4-3) opens, allowing you to create a new project or open an existing project.

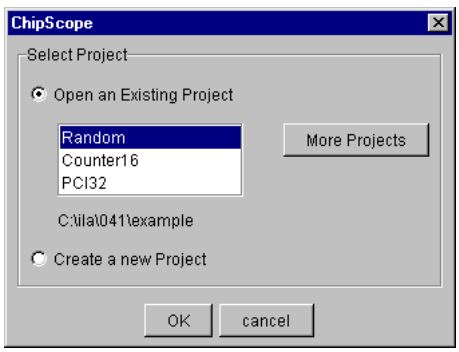

Figure 4-3: **Selecting a Project**

Creating A New Project

To create a new project, select **Create a new Project** (Figure 4-3) and click **OK**. In the **Choose New Project File** dialog box (Figure 4-4), enter a new project filename name (using the .cpj extension) and click **Open**.

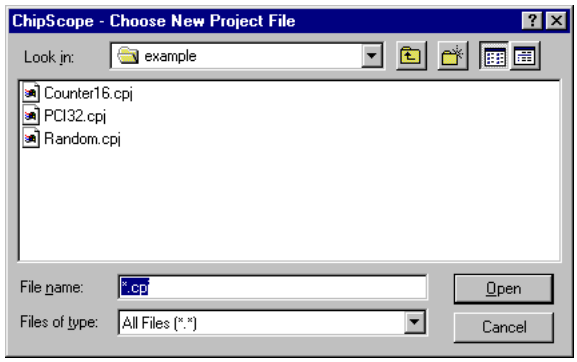

Figure 4-4: **Creating a New Project**

Opening An Existing Project

To open an existing project, select it from the list of recently opened projects. To browse through all available project files, select **More Projects**. When you locate the desired project click **Open**.

Saving Projects

Projects are automatically saved when you exit the ChipScope software.

To rename the current project, or to save a copy to another filename, select **File** → **Save Project As** (Figure 4-5), type the new name in the dialog box, and click **Save**.

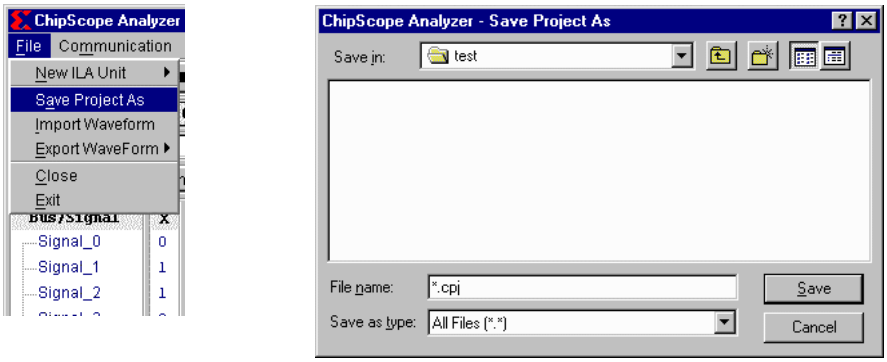

Figure 4-5: **Saving a Project**

Importing and Exporting

Only VCD waveforms can be imported. You can import the waveform display from a VCD file and export to VCD, FBDF, and ASCII files. The VCD format is a common waveform viewer file format and the FBDF format is compatible with the Agilent Technologies' 16700 Series logic analyzers. The ASCII format is a text-only list format that is well suited to a script parser or spreadsheet import.

To import a waveform, select **File** → **Import Waveform**. A dialog box opens, allowing you to browse for waveform files. After locating and selecting the desired file, click **Open**. The waveform display in the ChipScope window now contains the imported waveform.

You can export a waveform file in a similar fashion. To export the waveform to a VCD file, select **File** → **Export Waveform** → **VCD Export**. To export the waveform to a FBDF file, select **File** → **Export Waveform** → **FBDF Export**. To export the waveform to an ASCII file, select **File** → **Export Waveform** → **ASCII Export**. In each case, a dialog box opens and you can browse for the desired storage folder location. After finding the target location and entering the waveform filename, click **Save**. The waveform is now stored in the desired format.

Closing and Exiting ChipScope

To close a ChipScope window, select **File** → **Close**. To exit the ChipScope program, select **File** → **Exit**. In both cases, if you have not stored the waveform, a dialog box opens, asking if you want to store the current waveform before closing/exiting. If you select **Yes**, another dialog box opens, from which you can save the waveform.

The difference between closing and exiting is: closing closes the current ChipScope window; exiting closes all ChipScope windows and ends the program. However, if only one ChipScope window is open, closing and exiting have the same effect.

Opening and Closing a MultiLINX Connection

You can connect the MultiLINX cable to the host computer by using a serial communications port (e.g., COM1), the USB (Universal Serial Bus) port connection, or both. However, only one connection to the MultiLINX cable is supported at a time.

Opening a Serial Port MultiLINX Connection

If the MultiLINX cable connects to the host computer by way of the serial port, then select **Communication** → **Open Serial Port** (Figure 4-6).

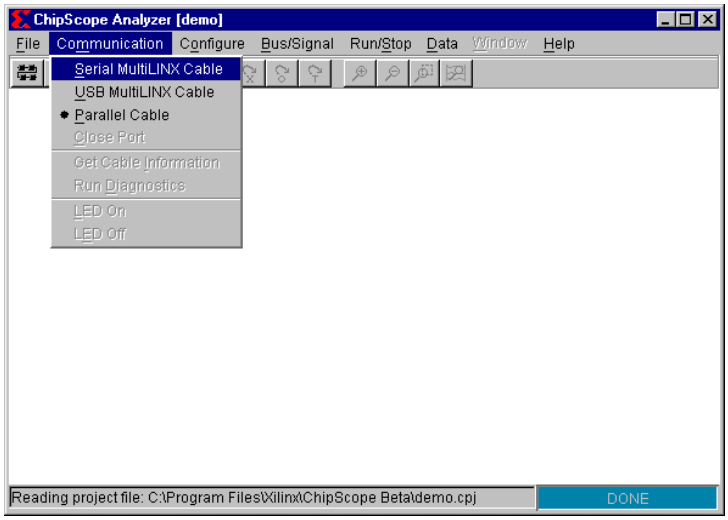

Figure 4-6: **Opening a Serial Port Connection to MultiLINX**

After you select **Communication** → **Open Serial Port**, a small dialog box opens (Figure 4-7). Enter the proper serial port name, select the baud rate for the serial port connected to the MultiLINX cable, then click **OK**. Make sure you select a port that is not in use by another resource.

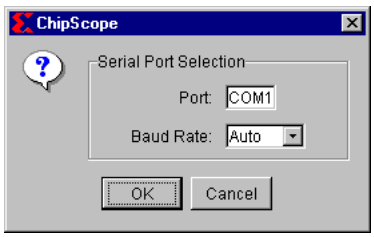

Figure 4-7: **Selecting a Serial Port**

When the connection opens, a success message appears in the status bar at the bottom of the ChipScope window. At this point, ChipScope queries the Boundary Scan chain to determine its composition (see Figure 4-11, page 4- 7). If the MultiLINX connection fails to open, a dialog box opens, notifying you of the problem.

Opening a USB Port MultiLINX Connection

If the MultiLINX cable connects to the host computer by way of the USB port, then select **Communication** → **Open USB Port** (Figure 4-8).

When the connection opens, a success message appears in the status bar at the bottom of the ChipScope window. At this point, ChipScope queries the Boundary Scan chain to determine its composition (see **Setting Up the Boundary Scan Chain**, page 4- 6). If the MultiLINX connection fails to open, a dialog box opens, notifying you of the problem.

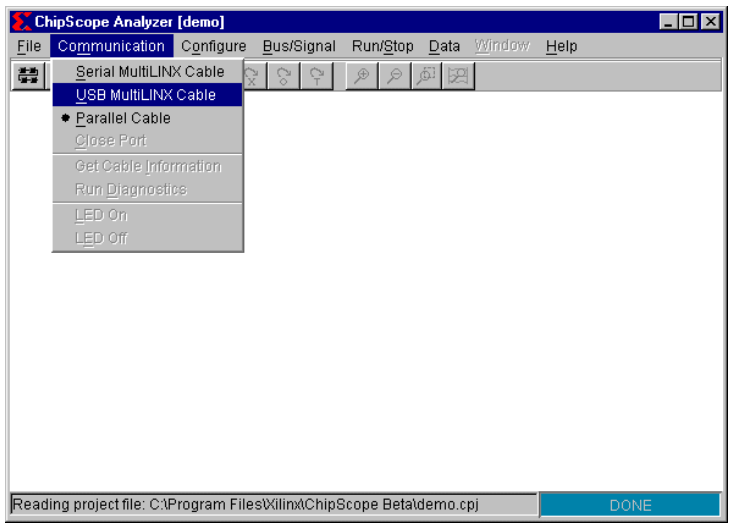

Figure 4-8: **Opening a USB Connection to MultiLINX**

Closing the MultiLINX Connection

Select **Communication** → **Close Port** to close the connection to the MultiLINX cable. You must re-establish a connection to the MultiLINX cable before any communication between the ChipScope program and the ILA units in the target device can resume.

Getting MultiLINX Cable Information

You can upload information pertaining to the MultiLINX cable (such as the hardware version, memory size, etc.) by selecting **Communication** → **Get Cable Information**. This menu option will also flash the LED on the MultiLINX cable five times and will report

MultiLINX cable diagnostic information. See the sample dialog in Figure 4-9 for the MultiLINX cable information.

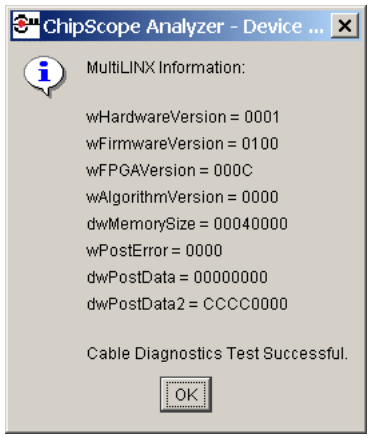

Figure 4-9: **Getting Cable Information**

Opening a Parallel Cable Connection

ChipScope supports the Parallel Cable III and Parallel Cable IV. To open a connection to the Parallel Cable, make sure the cable is connected to one of the computer's parallel ports. Select **`Communication`** → **`Parallel Cable`** (Figure 4-10). ChipScope prompts you for the port name. Type the printer port name in the port selection box (usually the default LPT1 is correct) and click **`OK`**. If successful, ChipScope queries the Boundary Scan chain to determine its composition (see **Setting Up the Boundary Scan Chain**, page 4- 6).

If ChipScope returns the error message "Failed to Open Communication Port", verify that the Parallel Cable III or Parallel Cable IV is connected to the correct LPT port. If you have not installed the Parallel Cable III or Parallel Cable IV driver, follow the instructions in Note 2, page 1- 10 in Chapter 1 to install the required device driver software.

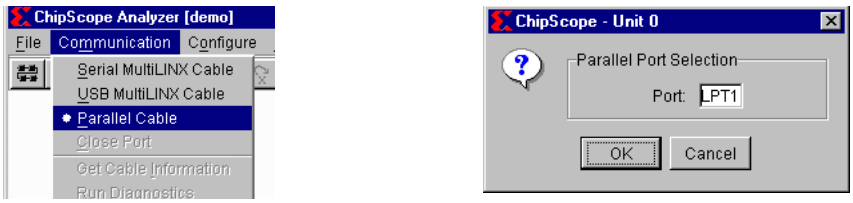

Figure 4-10: **Opening a Parallel Cable III Connection**

Configuring the Target Device(s)

You can use ChipScope software with one or more target devices (Virtex, Virtex-E, Virtex-II, Virtex-II Pro, Spartan-II or Spartan-IIE device families. The first step is to set up all of the devices in the Boundary Scan chain.

Setting Up the Boundary Scan Chain

Once ChipScope has successfully communicated with a download cable, it automatically queries the JTAG chain to find its composition. All Xilinx Virtex/E/EM/II, Spartan-II/IIE, Spartan-XL, 9500/XL/XV, 4000XL/XLA, 18V00, CoolRunner™, CoolRunner-II, and System ACE™ devices are automatically detected. The entire IDCODE can be verified for Virtex, Virtex-E, Virtex-EM, Virtex-II, Virtex-II Pro, Spartan-IIE, and Spartan-II devices. To view the chain composition, select **`Configure`** → **`Boundary Scan Setup`**. A dialog appears with all detected devices in order. For devices that are not automatically detected, the IR (Instruction Register) length must be specified to insure proper communication to

the ILA and ICON cores This information can be found in the device's BSDL file. The following example has a System ACE controller, XCV50 Virtex FPGA, and XC2V1000 Virtex-II FPGA in a chain (see Figure 4-11). UserID's can be read out of the ILA target devices (only the XCV50 and XC2V1000 devices in this example) by selecting **`Read User IDs`**.

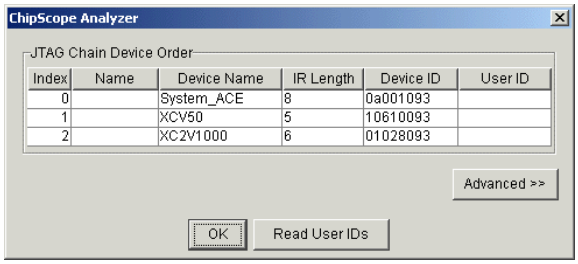

Figure 4-11: **Boundary Scan Setup Window**

The ChipScope Analyzer tool automatically keeps track of the TAP state of the devices in the Boundary Scan chain, by default. If the ChipScope Analyzer is used in conjunction with other Boundary Scan controllers (such as the System ACE CF controller or processor debug tools), then the actual TAP state of the target devices can differ from the tracking copy of the ChipScope Analyzer. In this case, the ChipScope Analyzer should always put the TAP controllers into a known state (e.g., the Test Logic Reset state) before starting any Boundary Scan transaction sequences. Clicking on the Advanced button on the Boundary Scan Setup dialog box reveals the parameters that control the start and end states of Boundary Scan transactions (see Figure 4-12). Use the second parameter if the Boundary Scan chain is shared with other Boundary Scan controllers.

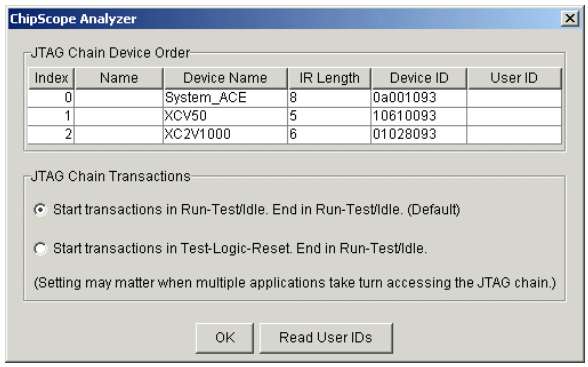

Figure 4-12: **Advanced Boundary Scan Parameters Setup Window**

Device Configuration

The ChipScope Analyzer is able to configure target FPGA devices using MultiLINX, Parallel Cable III, or Parallel Cable IV cables in JTAG mode only.

Note: Slave Serial configuration mode is no longer supported.

If the target device is to be programmed using the MultiLINX or Parallel Cable III download cable by way of the JTAG port, select **`Configure`** → **`JTag Mode`** (Figure 4-13).

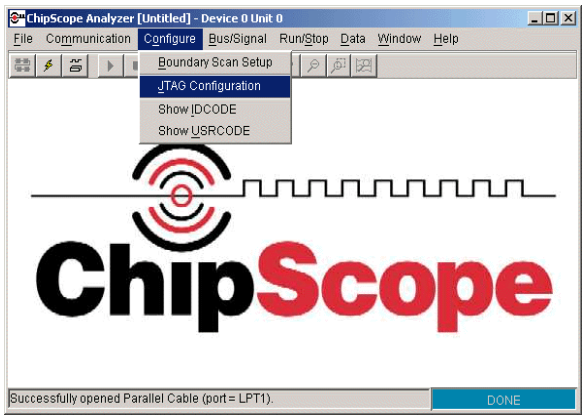

Figure 4-13: **Configuring JTAG Mode Window**

After selecting the configuration mode, the Configuration Selection dialog box (Figure 4-14) opens. This dialog reflects the configuration choice, and defaults to a blank entry for the configuration file. ChipScope supports MCS, BIT, and RBT files as inputs.

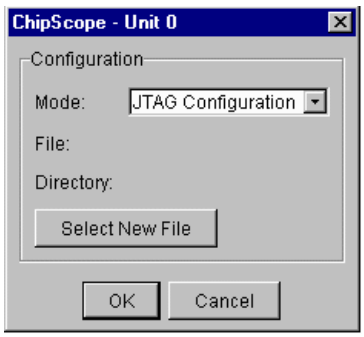

Figure 4-14: **Selecting a Part**

To select the BIT file to download, click on **Select New File**. The Open Configuration File dialog box (Figure 4-15) opens. Using the browser, select the device file you want to use to configure the target device. It is important to select a BIT file generated with the proper BitGen settings.

For example, if the target device is configured using the JTAG port, then use the BitGen option **-g StartupClk:JtagClk** when creating the configuration file. Once you locate and select the proper device file, click **Open** to return to the Configuration Selection dialog.

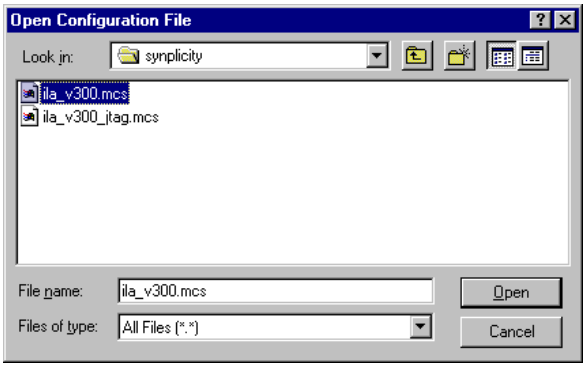

Figure 4-15: **Opening a Configuration File**

Once the mode and BIT file have been chosen, click **OK** to configure the device.

Observing Configuration Progress

While the device is being configured, the status of the configuration is displayed at the bottom of the ChipScope window. If using the MultiLINX cable, first the cable is initialized for JTAG configuration download. Next, the progress of the bitstream download is displayed. If the DONE status is not displayed, a dialog box opens, explaining the problem encountered during configuration. If the download is successful, the target device is automatically queried for the ILA core, and the Trigger Setup toolbar is displayed.

Displaying JTAG ID Codes

One method of verifying that the target device was configured correctly is to upload the device and user-defined ID codes from the target device.

For instance, to upload and display the user-defined ID code (i.e., the 8-digit hexadecimal code that can be set using the BitGen option **-g UserID**), select **Configure** → **Show USERCODE** (Figure 4-16). Use **Configure** → **Show IDCODE** to display the fixed device ID code.

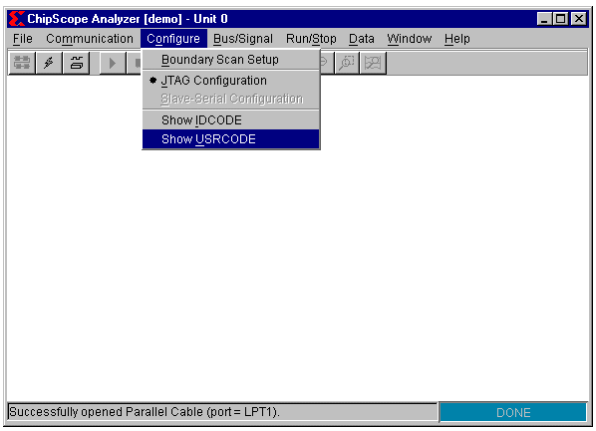

Figure 4-16: **Uploading User-Defined ID Code**

The ID codes are displayed in a small dialog box (Figure 4-17).

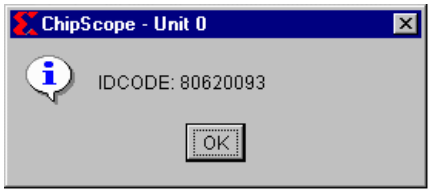

Figure 4-17: **Viewing User-Defined ID Code**

Opening the Trigger Setup Toolbar

To set up the trigger for a device that has already been configured with a design containing an ILA core, select **`Data`** → **`Trigger Setup`** (Figure 4-18).

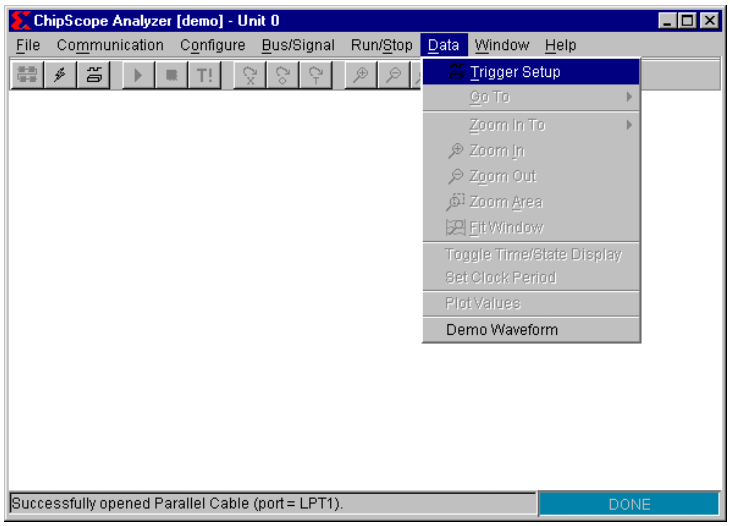

Figure 4-18: **Opening the Trigger Setup Toolbar**

After you select **`Trigger Setup`**, the ChipScope program queries the ILA unit to determine the proper settings for the trigger setup. If the device has just been configured, then it is queried and the Trigger Setup toolbar appears automatically.

The parameters associated with the ILA unit appear at the bottom of the ChipScope Analyzer window (see Figure 4-19 for example of Trigger Width: 8, Signal Count: 16, Data Depth: 2048, ILA Core Version 1.1 extended match units). The Trigger Width, Signal Count, Data Depth, and Extended Features parameters correspond to the ILA core parameters used when the core is generated.

Figure 4-19: **Viewing the Trigger Setup Toolbar**

Setting Up the Trigger

The trigger mechanism inside each ILA core can be modified at run-time without having to re-compile the design. The following sections describe how to modify the various components that make up the trigger mechanism.

Basic Match Unit Comparison Values

The match units are called **M*n*** (where ***n*** is 0 or 1, depending on the number of trigger match units in the ILA core) and can be one of two types: basic or extended. If the match units are basic, then you can change only the comparison value (Figure 4-20). You can set each bit of the match unit to one of the following values:

- X : Any value (logical zero or logical one)
- 0 : Logical zero only
- 1 : Logical one only
- R : Rising edge only
- F : Falling edge only
- B : Both edges (rising edge or falling edge)

You can set match word values by clicking on each bit until the desired value appears, or by clicking on the bit value and typing the desired value. Moving the cursor over (but not clicking on) a bit in the match word causes a tool tip window to appear that indicates the bit position and current match value.

Note: If you have imported signal names using the Bus/Signal->Import Signal Names menu option, then the tool-tip for each match unit value bit will display the name of the net that is attached to the corresponding ILA trigger input in addition to the bit position and current match value.

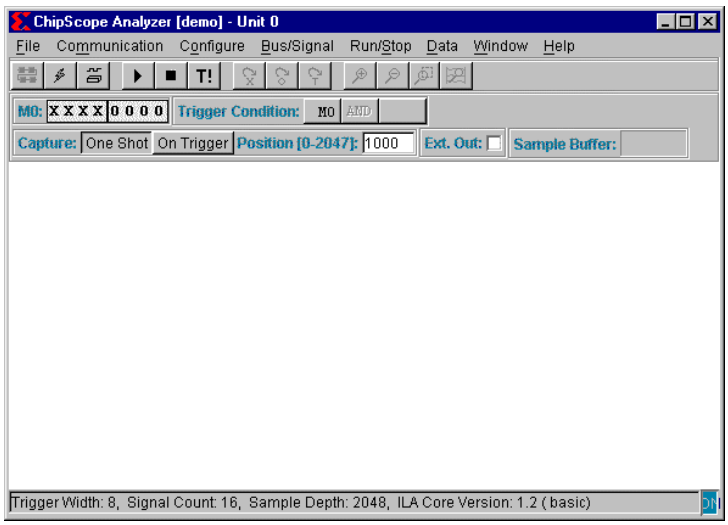

Figure 4-20: **Setting the Basic Match Comparison Value**

Extended Match Unit Comparison Value

If the match unit **M*n*** (where ***n*** is 0 or 1, depending on the number of trigger match units in the ILA core) is an extended match unit, then you can configure both the comparison type and value during trigger setup (Figure 4-21).

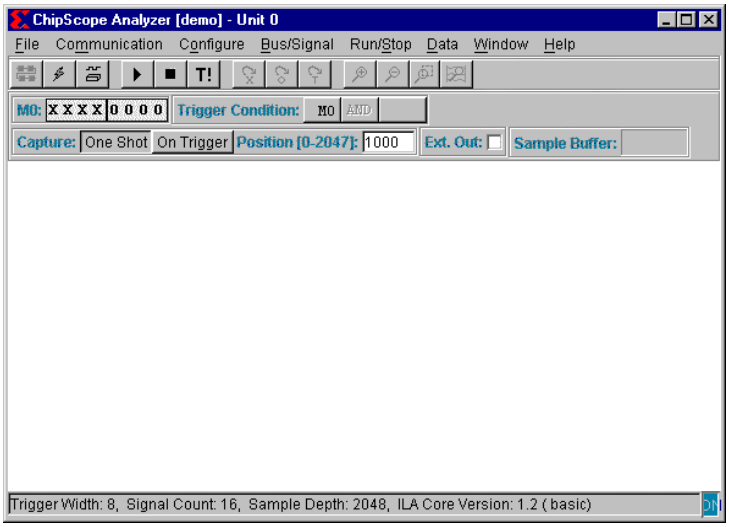

Figure 4-21: **Setting the Extended Match Comparison Type and Value**

The possible comparison types are:

- = : equal to
- <> : not equal to
- > : greater than
- >= : greater than or equal to
- < : less than
- <= : less than or equal to

The possible match comparison bit values depend on the comparison type. For instance, when the comparison type is set to '=', you can set each bit of the comparison value to one of the following:

- X : Any value (logical zero or logical one)
- 0 : Logical zero only
- 1 : Logical one only
- R : Rising edge only
- F : Falling edge only
- B : Both edges (rising edge or falling edge)

If the comparison type is set to something other than "=", then you can set each bit of the comparison to one of the following:

- 0 : Logical zero only
- 1 : Logical one only

You can set the match word values by clicking on each bit until the desired value appears, or by clicking on the bit value and typing the desired value. If you move the cursor over a bit in the match word and do not click, a tool tip dialog box opens displaying the bit position and current match value.

Setting Up Pulse Width and Event Count

You can configure an extended match unit to find matches that occur over one or more clock cycles contiguously (in a row) or non-contiguously (not necessarily in a row). If the match conditions must remain satisfied during a specific number of contiguous clock cycles, then the match **Pulse Width Duration** must be measured. If it only needs to occur for a specific number of clock cycles (not necessarily in a row), then the **Match Event Count** must be measured.

To measure the **Pulse Width Duration**, click **Width** (Figure 4-22). The match condition is true only if the match condition value is detected for at least ***n*** clock cycles in a row (where ***n*** can be any value from 1 to 65,536). For a subsequent occurrence of the match condition to become satisfied, the match condition "pulse" must return to its non-active state in order to reset the **Pulse Width Duration** detection logic.

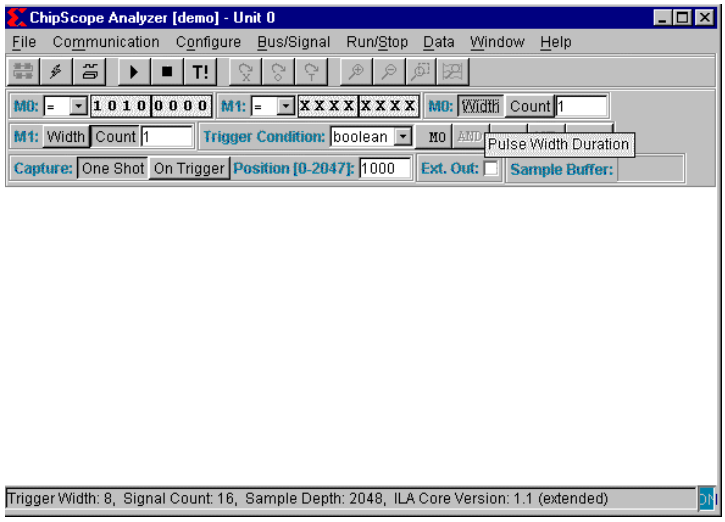

Figure 4-22: **Selecting the Pulse Width Duration**

To measure the **Trigger Event Count**, click **Count** (Figure 4-23). The match condition is true only if the match condition value is detected for at least ***n*** clock cycles *not necessarily* in a row (where ***n*** can be any value from 1 to 65,536). The match condition **Trigger Event Count** automatically resets after the match condition is satisfied.

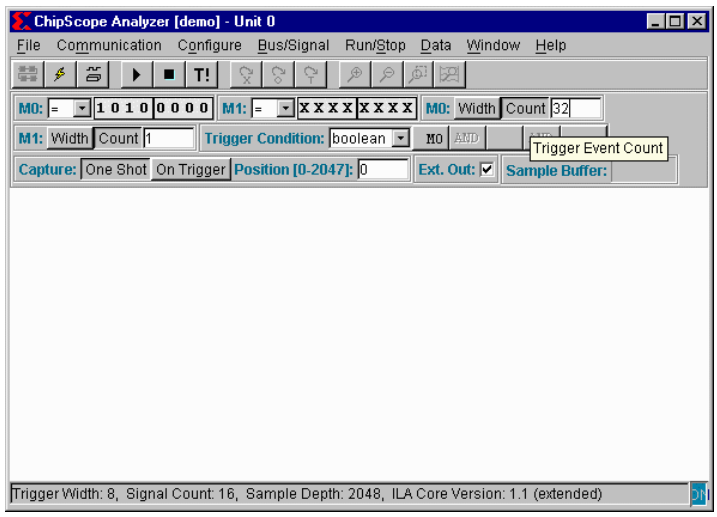

Figure 4-23: **Selecting the Trigger Event Count**

Setting Up a Boolean Trigger Condition

The **Trigger Condition** field describes how the match units (M0, M1) and the external trigger input (EXT) can be combined to form an overall trigger condition. Both basic and extended trigger match units can build boolean equations from the available match units and the external trigger input to form the overall trigger condition. Select the boolean trigger condition type by choosing **Boolean** from the pull-down menu (not available in the "basic" trigger match type case).

To designate the equation, click the buttons to the immediate right of the **Trigger Condition** field in the trigger setup toolbar. For instance, if you want the trigger condition boolean equation to be "not M0 or EXT" (Figure 4-24), do the following:

- Click twice on the first button to select **!M0** (i.e., **not M0**).
- Click once on the third button to select **EXT**
- Click once on the middle button that reads **AND** to select **OR**

Clicking the button until it is blank effectively removes the match condition or external trigger input out of the boolean equation.

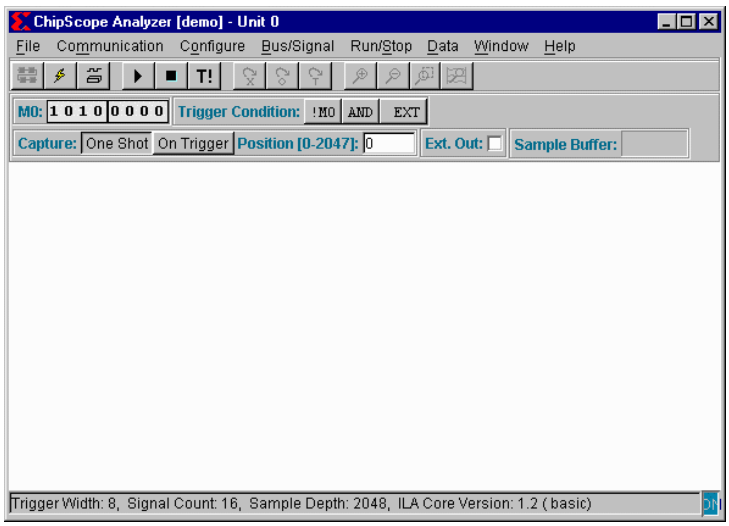

Figure 4-24: **Selecting the Boolean Trigger Condition**

Setting Up a Macro Trigger Condition

Only the extended trigger match units can build "if-then" macro equations out of the available match units and the external trigger input to form the overall trigger condition. Select the macro trigger condition type by selecting the "macro" type from the pull-down menu. An example of the "if-then" macro condition is "if M0 is satisfied, and then ~EXT is satisfied, then the overall trigger condition is satisfied."

To designate the macro equation, click on the buttons to the immediate right of the **Trigger Condition** field in the trigger setup toolbar. For instance, if you want the trigger condition boolean equation to be "M0 then not EXT" (Figure 4-25), do the following:

- Click the third button four times to select **!EXT** (i.e., **not EXT**).
- Click the first button one time to select **M0**
- The middle button is automatically set to **THEN**

Clicking the first and/or third button until it is blank effectively removes the match condition or external trigger input from the macro equation. However, use the boolean equation if neither condition in the macro equation is used.

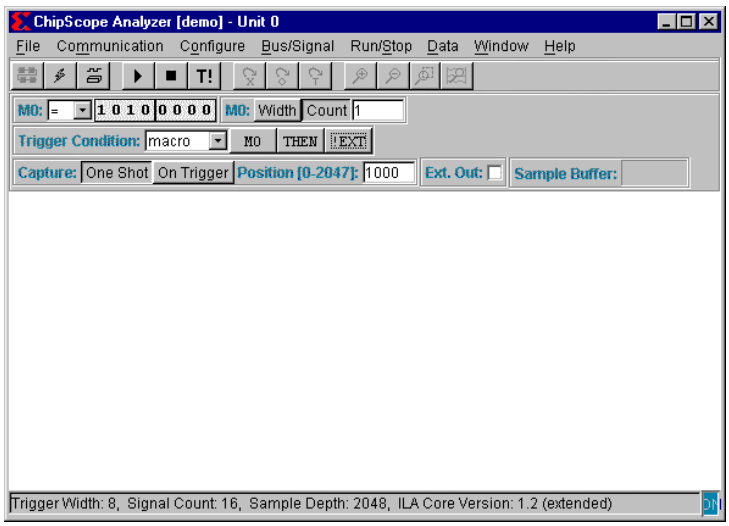

Figure 4-25: **Selecting the Macro Trigger Condition**

Selecting One Shot Capture Mode

The One Shot capture mode captures an entire sample buffer of data once the trigger condition is satisfied. This capture method allows you to capture data both before and after the trigger condition is satisfied. Click **One Shot** in the trigger toolbar (Figure 4-26) to select the One Shot capture mode.

During the One Shot capture mode, you can set the trigger position to anywhere from the beginning of the capture buffer (Position = 0) to the end of the capture buffer (Position = Sample Depth -1). To set the trigger position, click inside the box to the right of the **Position** setting on the trigger toolbar, (Figure 4-26). For example, if the trigger position is set to 1000, then the 1000 data samples captured before the trigger condition is satisfied are displayed along with 1047 samples captured after the trigger.

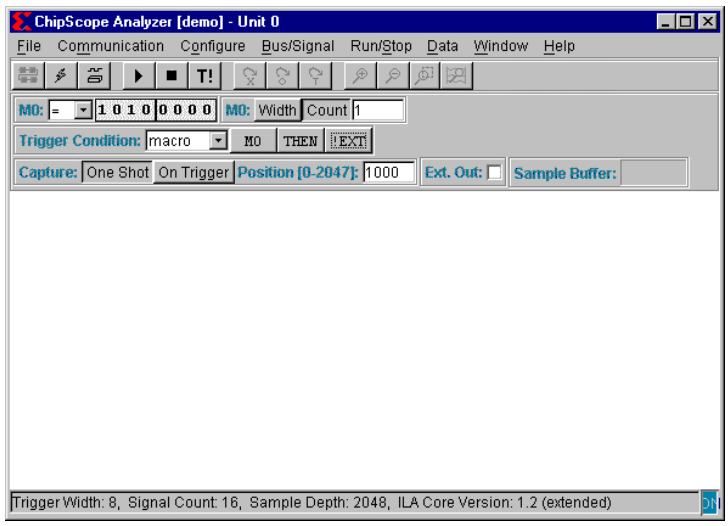

Figure 4-26: **Selecting One Shot Capture Mode**

Selecting On Trigger Capture Mode

The On Trigger capture mode continues capturing data after each trigger condition is satisfied, until the entire sample buffer is full. You can set the number of samples captured per trigger to any value from 1 to 16. To select this mode, click **On Trigger** in the trigger toolbar (Figure 4-27). Select the number of samples per trigger from the **Samples per Capture** scrolling list box on the trigger toolbar (Figure 4-27).

For example, if you select the trigger position 8, then 8 data samples are captured after each occurrence of the trigger condition until the ILA unit's capture storage resources are completely full. In some cases, the last occurrence of the trigger condition might result in fewer samples stored, depending on the value of the **Samples per Capture** setting and the sample depth of the ILA unit.

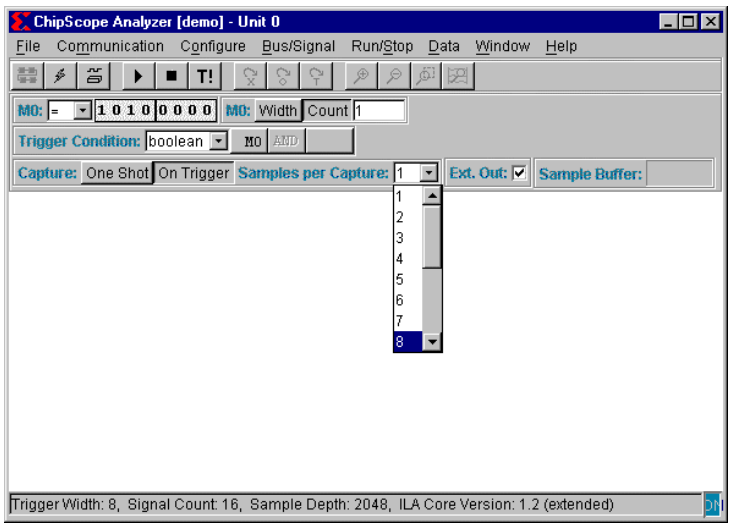

Figure 4-27: **Selecting On Trigger Capture Mode Window**

Enabling External Trigger Output

Each ILA unit can drive out its overall trigger condition as an external trigger output to the ICON unit. There it is logically OR'ed with the external trigger output of all the other ILA units in the target device. However, you can disable the ILA unit from driving out its overall trigger condition (and drive a logical "0" instead) by deselecting the **Ext. Out** field in the trigger setup toolbar. Selecting the **Ext. Out** field enables the ILA unit's individual external trigger output (Figure 4-28).

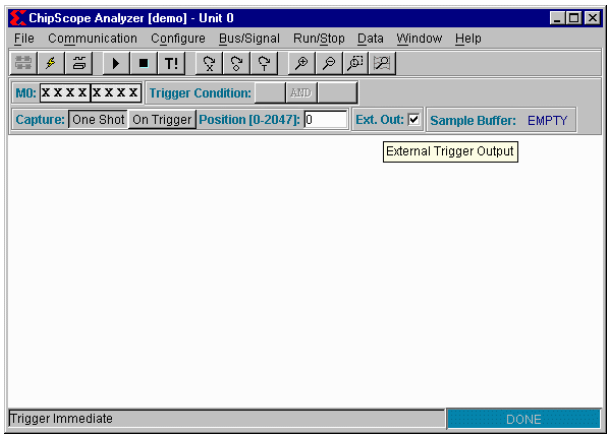

Figure 4-28: **Enabling the External Trigger Output**

Capture Status Window

After you arm the trigger by selecting **Run/Stop** → **Run**, the status of the ILA unit appears in the **Sample Buffer** display area with one of the following values:

- ARMED : The trigger is currently armed and waiting for an occurrence of the trigger condition.
- <value> : A number <value> between 1 and Sample Depth - 1 that represents how many data samples have currently been captured.
- FULL : The capture storage is full and the capture process has ended.
- STOPPED : The ILA unit has been halted by the act of stopping the acquisition.

During One Shot capture mode, once the ILA unit progresses from the ARMED state to the FULL state, ChipScope program automatically displays the captured data in the waveform window (Figure 4-29). Note that the X-axis of the waveform is measured in number of data samples. Data sample 0 is always at the location of the trigger condition.

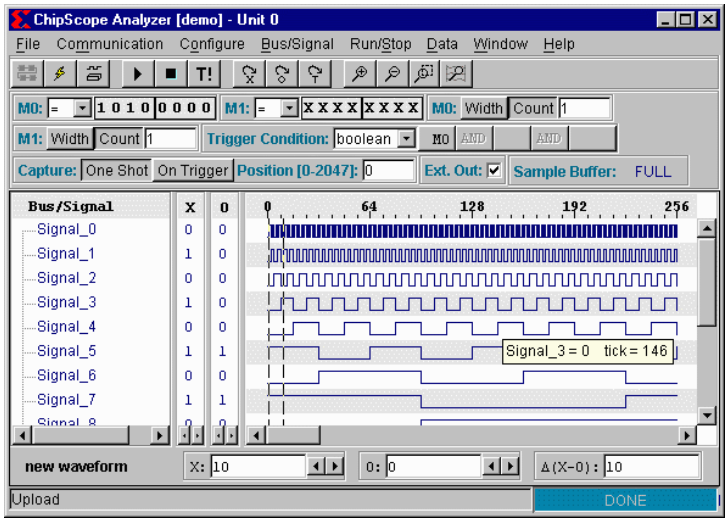

Figure 4-29: **Viewing the One Shot Capture Mode Waveform**

During On Trigger capture mode, once the ILA unit progresses from the ARMED state to the FULL state, the ChipScope program automatically displays the captured data in the waveform window (Figure 4-30). Data sample 0 is always at the location of the *first* trigger condition.

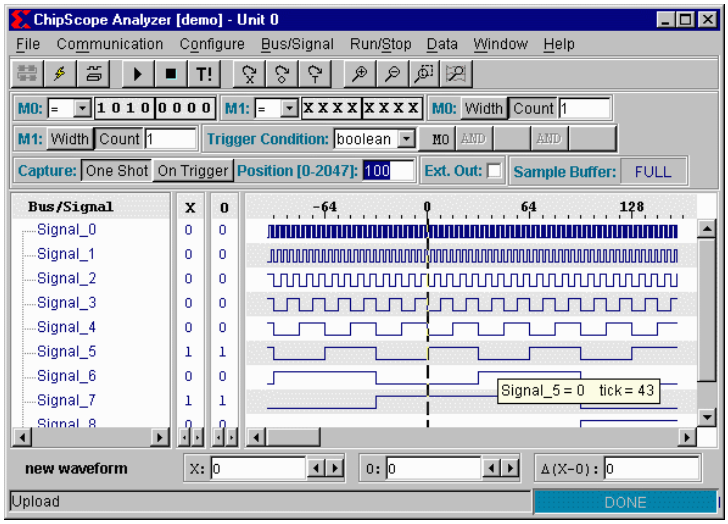

Figure 4-30: **Viewing the On Trigger Capture Mode Waveform**

Running and Stopping the Trigger

Running/Arming the Trigger

After setting up the trigger, select **Run/Stop** → **Run** to arm it. The trigger stays armed until the trigger condition is satisfied or the user disarms the trigger. Once the trigger condition is satisfied, the trigger automatically disarms and the captured data appears in the waveform window.

To force the trigger, select **Run/Stop** → **Trigger Immediate**. This causes the ILA unit to ignore the trigger condition and trigger immediately. After the sample buffer fills with data, the trigger disarms and the captured data appears in the waveform window.

Stopping/Disarming the Trigger

To disarm the trigger, select **Run/Stop** → **Stop Acquisition**. If the trigger condition has been satisfied at least once before the acquisition is stopped, the ChipScope program disarms the trigger and displays the captured data. Subsequent selections of **Run/Stop** → **Run** cause the trigger to re-arm.

Using Buses and Signals

Grouping Signals Into a Bus

You can group up to 64 signals to form a bus. Hold down the shift key and use the mouse to select one or more ungrouped signals in the **Bus/Signal** display, then select **Bus/Signal** → **Group Into Bus**. When the text input box opens, enter a name for the

new bus, then click **Enter**. Names must be unique and may contain only letters, numbers, and underscores (Figure 4-31).

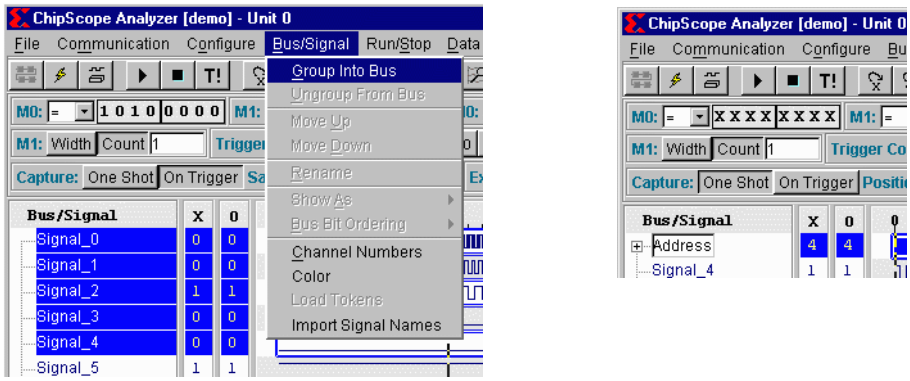

Figure 4-31: **Grouping Signals into a Bus**

Ungrouping Signals From a Bus

To ungroup a signal from a bus, highlight a bus in the **Bus/Signal** display window, then select **Bus/Signal** → **Ungroup From Bus**. This removes the bus and places all its signals in the root level (Figure 4-32).

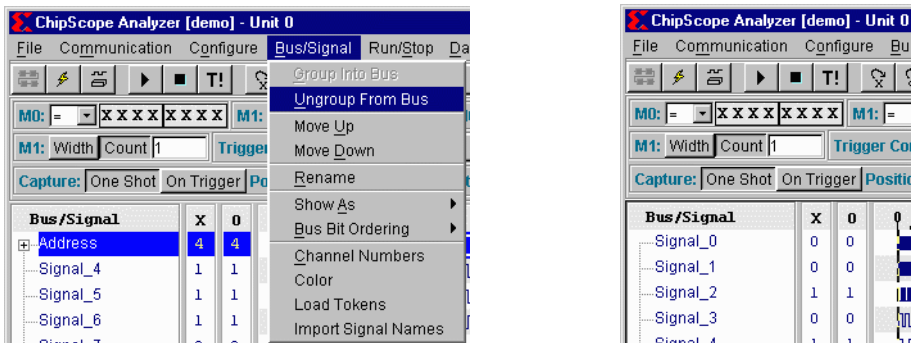

Figure 4-32: **Ungrouping Signals from a Bus**

Moving Buses and Signals

To move buses and signals up and down in the display, highlight a signal or bus, then select **Bus/Signal** → **Move Up** or **Bus/Signal** → **Move Down**. When the position of a signal in a bus is changed, the bus values are recalculated and the new values appear in the wave display window (Figure 4-33). Buses and signals can also be moved by dragging and dropping them in the left hand signal pane.

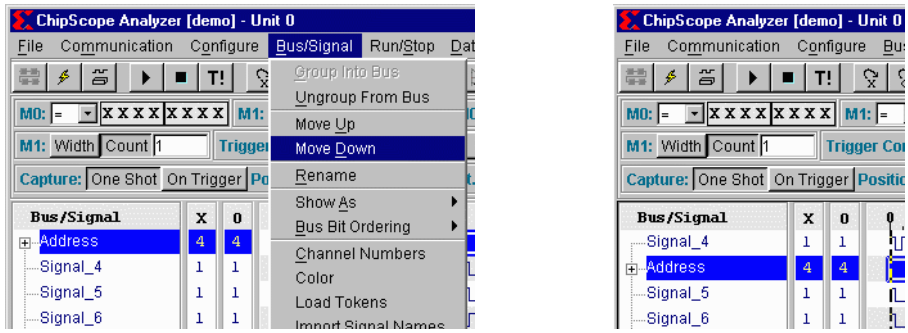

Figure 4-33: **Moving Buses and Signals**

Changing Bus and Signal Names

To rename buses and signals anytime for easy identification, click on the bus or signal to rename, then select **Bus Signal**→ **Rename**. A text input box (Figure 4-34) opens, prompting for the new name. Names must be unique and may contain letters, numbers, or underscores. Type the new name, then click **Enter**.

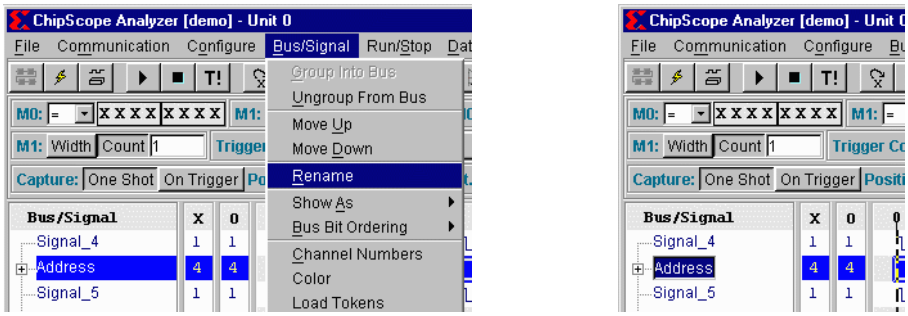

Figure 4-34: **Changing Bus and Signal Names**

Bus Radix Display

Buses can be configured individually to display different radixes in the wave display window. Available bus display options are: hexadecimal, binary, signed decimal, unsigned decimal, octal, ASCII, and Token. By default, bus values are displayed in hexadecimal. ASCII is only available when the bus specified is exactly 8 bits wide. To set the radix for a bus, select a bus in the **Bus/Signal** panel, then select **Bus/Signal** → **Show As**. The menu popup (Figure 4-35) allows you to choose from the available bus radix options.

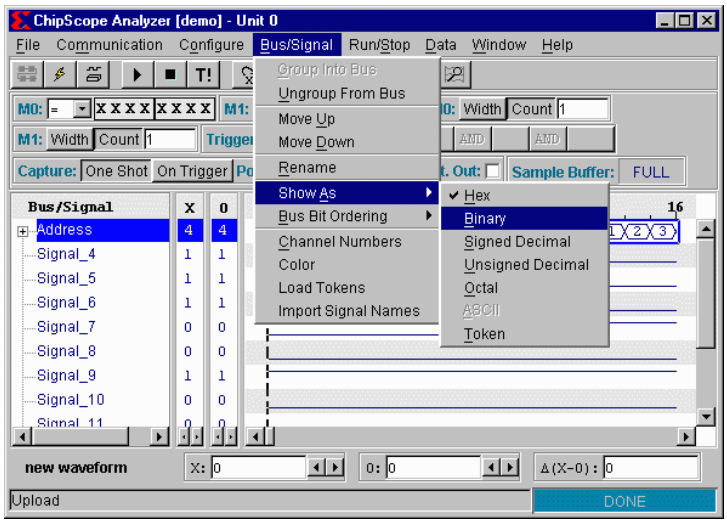

Figure 4-35: **Changing Bus Display Radix**

Using Tokens

Tokens are string labels that can be assigned to a particular bus value. These labels can be very useful in such applications as address decoding and state machines. Tokens are defined in a separate ASCII file, and loaded into the ChipScope Analyzer when appropriate. The token file itself (.tok extension) has a very simple format, and can be created or edited in any text editor. An example token file is provided in the token directory in the ChipScope install path (Figure 4-36).

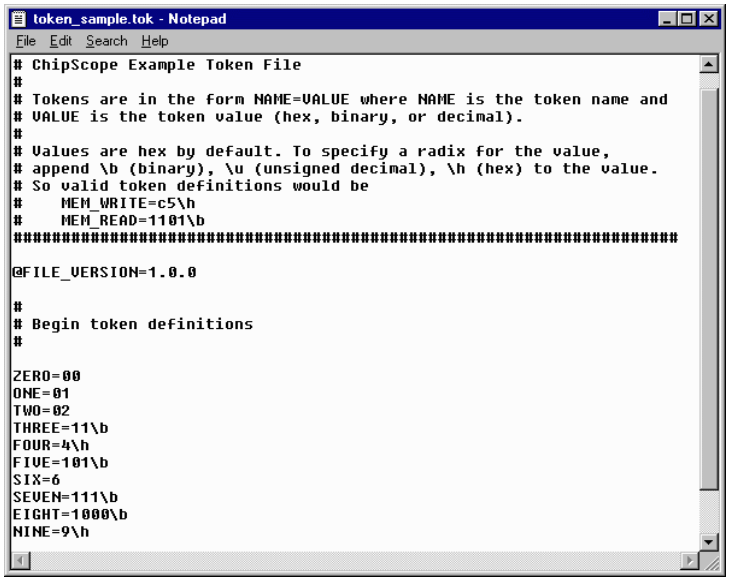

Figure 4-36: **Example Token File**

Tokens are chosen by selecting a bus, then choosing **`Bus/Signal`** → **`Load Tokens`**. A dialog opens and the user can choose the token file. Once the tokens are loaded, selecting **`Bus/Signal`** → **`Show As`** → **`Token`** enables the tokens for that particular bus. If the bus is wider than the tokens specify (such as choosing 4-bit tokens for an 8-bit bus) the upper bits are assumed 0 for the tokens to apply. Figure 4-37 shows such a waveform, with the example file in Figure 4-36 applied to a 5-bit bus.

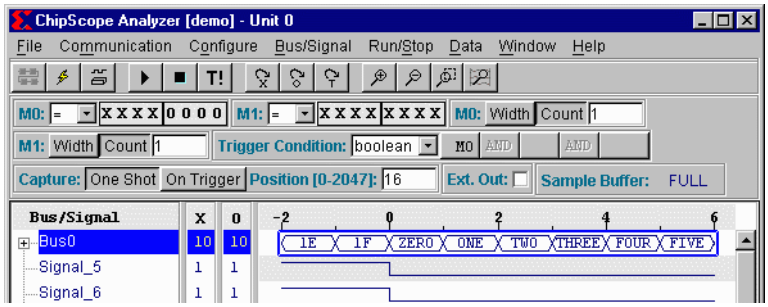

Figure 4-37: **Example Waveform with Tokens**

Bus Bit Ordering

The bits in each bus can be ordered top to bottom, or bottom to top, when computing their value in the wave display window. To place the most significant bit at the bottom, select **Bus/Signal** → **Bus Bit Ordering** → **Bottom(MSB) To Top(LSB)**. To place the most significant bit at the top, select **Bus/Signal** → **Bus Bit Ordering** → **Top(MSB) To Bottom(LSB)**. The default bit ordering is bottom to top (Figure 4-38).

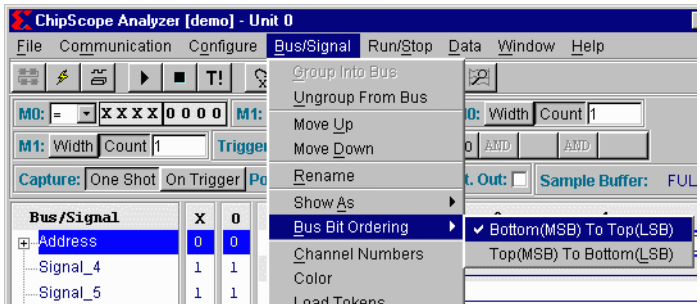

Figure 4-38: **Changing Bus Bit Ordering**

Signal Channel Number Display

Channel number display can be toggled to identify which of the ILA core data channels is connected to each of the signals in the waveform display. To toggle the display, select **Bus/Signal**→ **Channel Numbers**. By default, channel number display is turned off (Figure 4-39).

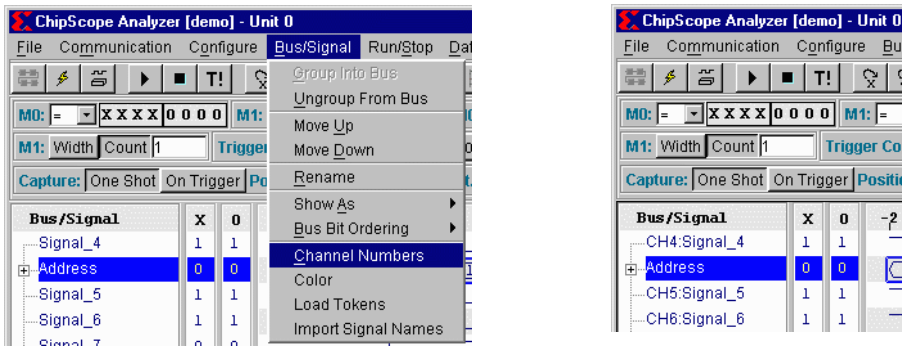

Figure 4-39: **Enabling Signal Channel Number Display**

Bus and Signal Coloring

Separate colors can be assigned to each bus and signal in the waveform display. To choose a color, highlight the signal or bus, then select **Bus/Signal** → **Color**. A color palette opens from which a discrete or customer color can be chosen in a variety of ways. A bus and its component signals can have different colors (Figure 4-40).

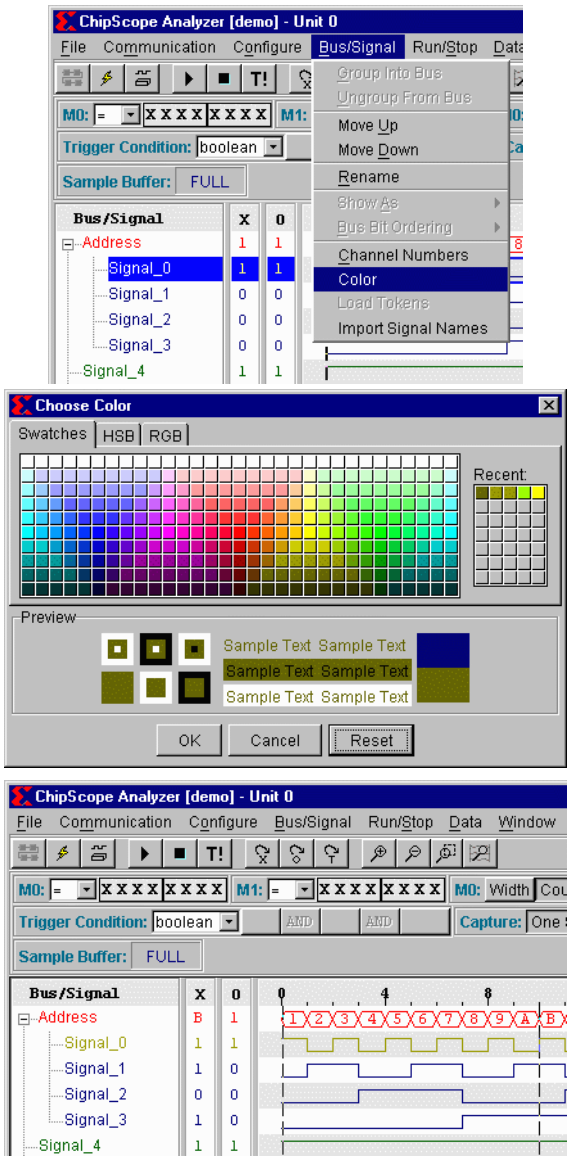

Figure 4-40: **Selecting a Waveform Color**

Signal Name Importing

Signal names can be directly imported from a ChipScope Core Inserter Project file. To import the names, select **Bus/Signal** → **Import Signal Names** and a dialogue box opens for navigating to the Core Inserter Project (**.cdc**) file. Select the file, then click **Open** to populate the waveform with the signal names from the project (Figure 4-41).

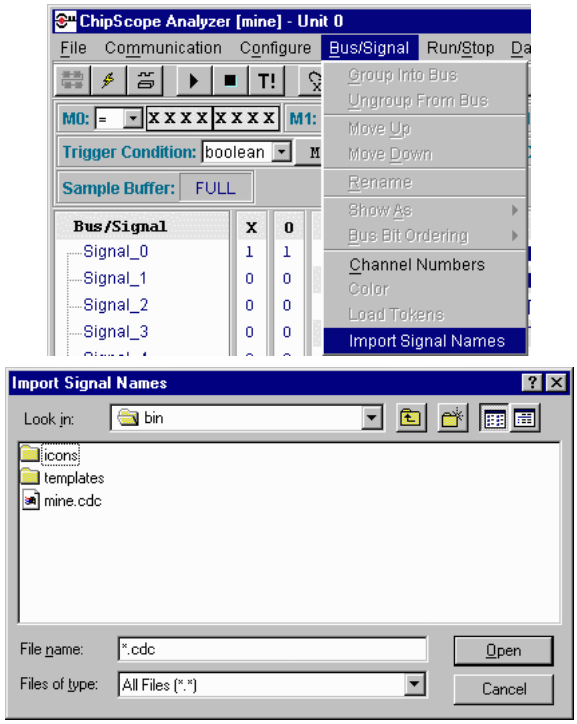

Figure 4-41: **Importing Signal Names**

Navigating the Waveform Window(s)

Centering the Waveform

Center the waveform display around a specific point in the waveform by selecting **Data** → **Go To**, then centering the waveform display around the **X** and **O** markers, as well as the first trigger position (Figure 4-42).

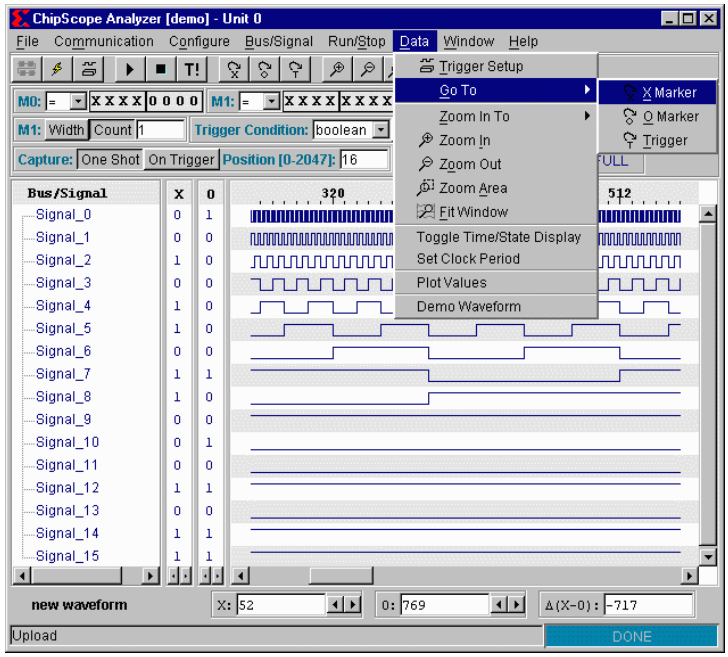

Figure 4-42: **Centering the Waveform Display**

Zooming In and Out

Select **Data** → **Zoom In** to zoom in to the center of the waveform display (Figure 4-43).

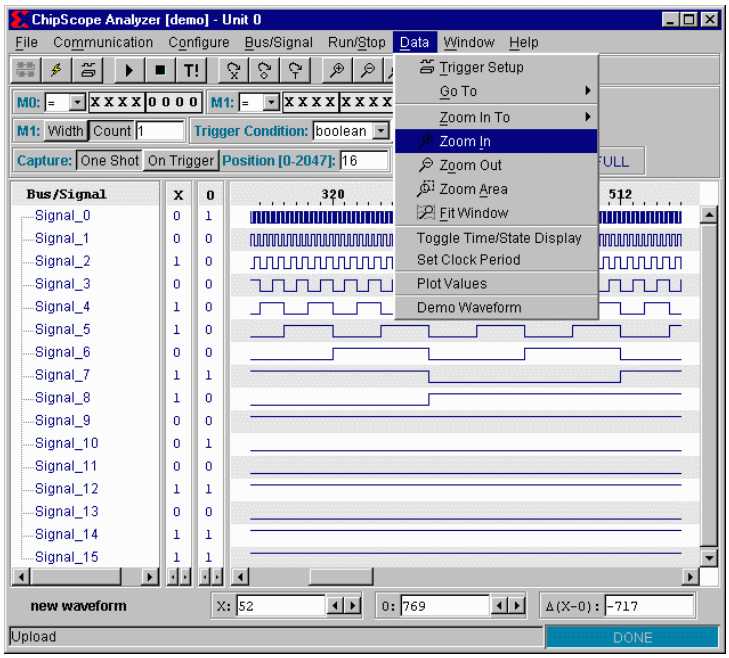

Figure 4-43: **Zooming in to the Center of the Waveform Display**

You can also zoom in to a specific place in the waveform. For example, select **Data** → **Zoom In To** → **X Marker** to zoom in to the **X** cursor location (Figure 4-44). Other zoom locations include the **O** cursor and first trigger position (that is, data sample 0).

To zoom in to a specific area of the waveform, select **Data** → **Zoom Area**, then click the left mouse button and drag to select an area of the waveform display. Using this method, the waveform display zooms in to the selected area.

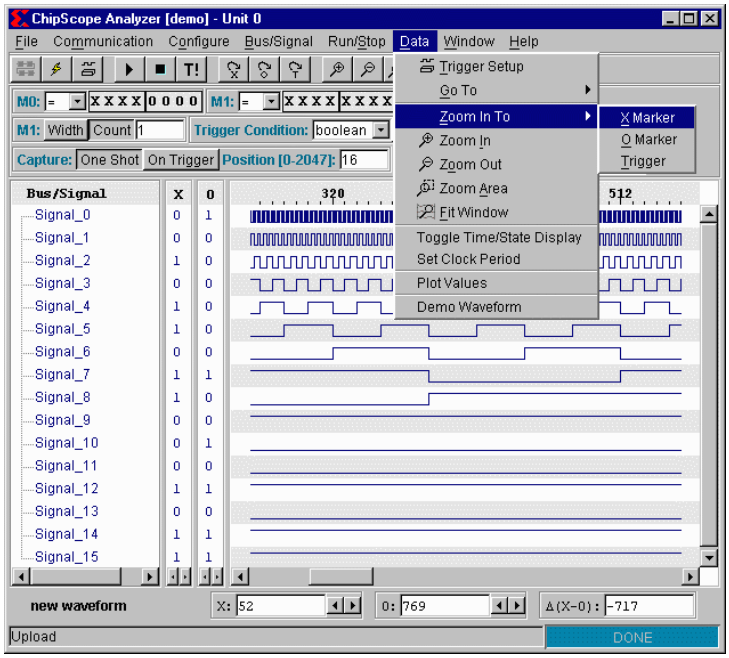

Figure 4-44: **Zooming in to the X-Marker of the Waveform Display**

To zoom out from a waveform, use **Data** → **Zoom Out** (Figure 4-45).

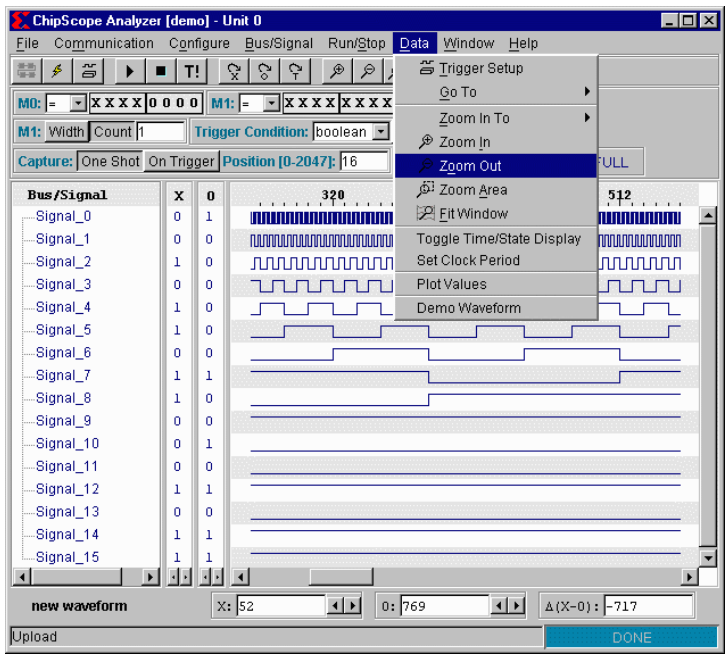

Figure 4-45: **Zooming out from the Waveform Display**

To view the entire waveform display select **Data** → **Fit Window** (Figure 4-46),

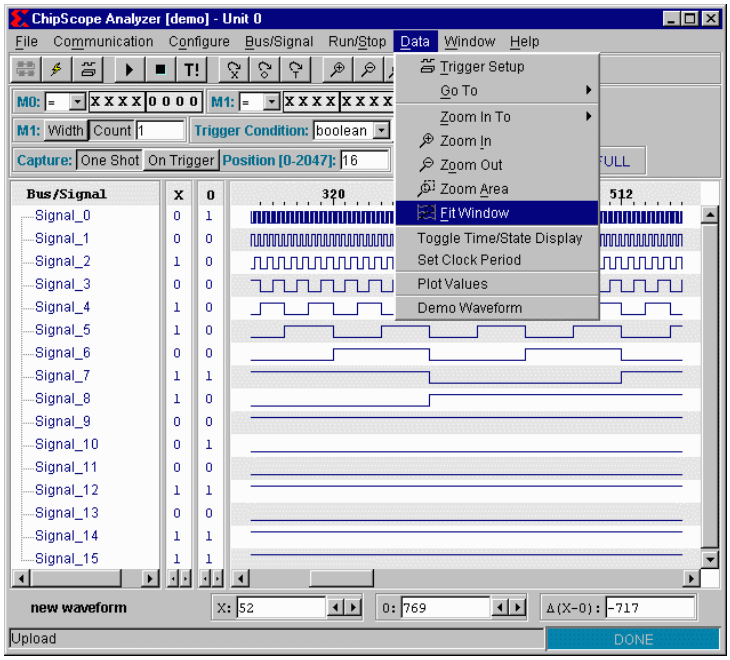

Figure 4-46: **Fitting the Waveform Display in the Window**

Toggling Time/State Display

The x-axis of the waveform can be displayed as the sample number relative to the trigger event (default) or by a time unit per sample starting at the first sample captured. By default, the time unit is 5ns/sample.

Setting a Sample Clock Period

When the waveform is viewed with the x-axis in time units, the sample clock period can be modified with this option. Units are always in ns.

Plot Values

This option brings up a separate window, with a bus or buses plotted in an x-y format. Two line types can be chosen: a scatter plot, which plots a single dot at each data point, and a line plot, which connects all the data points together with a line. Separate buses are displayed in separate colors (see Figure 4-47, with line graph chosen).

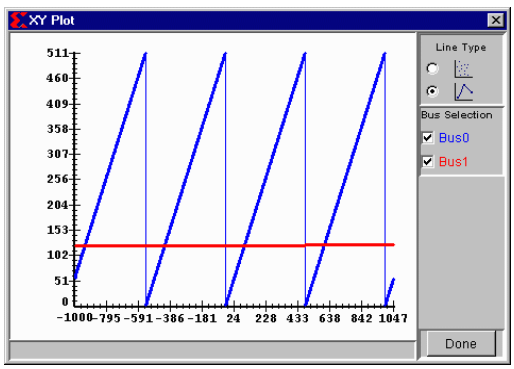

Figure 4-47: **Example of X-Y Plot**

Generating a Demo Waveform

To generate a sample waveform for trying out various ChipScope features without using actual hardware, select **Data** → **Demo Waveform**.

Changing Waveform Window Focus

If more than one ILA unit ChipScope window is open, use **Window** → **Unit** *n* (where *n* is the ILA unit number) to change focus between the windows.

Viewing the Help Pages

The ChipScope help pages contain only the currently opened versions of the ChipScope software and each of the ILA core units. Selecting **Help** → **About: ChipScope Software** displays the version of the ChipScope software. Selecting **Help** → **About: ILA Core** displays the versions of all of the open ILA units.

ChipScope Main Toolbar Features

In addition to the menu options, other ChipScope commands are available on a toolbar residing directly below the ChipScope menu (Figure 4-48).

Figure 4-48: **Main ChipScope Toolbar Display**

The toolbar buttons (from left to right) correspond to the following equivalent menu options:

- **Open Cable/Search JTAG Chain** : automatically detects the cable, and queries the JTAG chain to find its composition
- **Configure with Last Used Settings** : same as Configure → JTAG Configuration
- **Trigger Setup** : same as **Data** → **Trigger Setup**
- **Run** : same as **Run/Stop** → **Run** (F5)
- **Stop** : same as **Run/Stop** → **Stop** (F9)
- **Trigger Immediate** : same as **Run/Stop** → **Trigger Immediate** (Ctrl + F5)
- **Go To X Marker** : same as **Data** → **Go To** → **X Marker**
- **Go To O Marker** : same as **Data** → **Go To** → **O Marker**
- **Go To Trigger** : same as **Data** → **Go To** → **Trigger**
- **Zoom In** : same as **Data** → **Zoom In**
- **Zoom Out** : same as **Data** → **Zoom Out**
- **Zoom Area** : same as **Data** → **Zoom Area**
- **Fit Window** : same as **Data** → **Fit Window**